Objektkatalog für das Straßen- und Verkehrswesen

Datenbereitstellungs-Schnittstelle für OKSTRA-Daten auf Basis des OGC Web Feature Service

- *OKSTRA-WFS* -

Version:	2.4.1
Datum:	31.03.2016
Status:	abgeschlossen
Dateiname:	N0112.doc
Verantwortlich:	B. Weidner

OKSTRA-Pflegestelle **http://www.okstra.de/**

interactive instruments GmbH
Trierer Straße 70-72
53115 Bonn

Herr Bernd Weidner
Tel. 0228 91410 74
Fax 0228 91410 90
Email weidner@interactive-instruments.de

Im Auftrag von

Bundesanstalt für Straßenwesen
Referat V2
Brüderstraße 53
51427 Bergisch Gladbach

Herr Gerd Kellermann
Tel. 02204 43 526
Fax 02204 43 674
Email kellermanng@bast.de

0 Allgemeines

0.1 Inhaltsverzeichnis

0 Allgemeines ... 3
0.1 Inhaltsverzeichnis ... 3

1 Zweck des Dokuments (informativ) ... 7
1.1 Leserkreis ... 7
1.2 Kernaussagen des Inhalts ... 7
1.3 Änderungen gegenüber der Version 1.0.2 des Dokuments ... 7

2 Gegenstand (normativ) ... 9

3 Regelungen zur Konformität (normativ) ... 11

4 Normative Bezüge (normativ) ... 13
4.1 Filter Encoding ... 13
4.2 Web Feature Service ... 13
4.3 OKSTRA® ... 13
4.3.1 OKSTRA®-Datenmodell ... 13
4.3.2 OKSTRA®-Profile ... 13
4.3.3 Koordinatenreferenzsysteme ... 13
4.4 Authentifizierung ... 13

5 Glossar (normativ) ... 15

6 Überblick über OGC WFS (informativ) ... 17

7 Entscheidungsgrundlagen (teilweise normativ) ... 19
7.1 Anforderungen an Filterausdrücke ... 19
7.1.1 Die Filtersprache ... 19
7.1.2 Eigenschaftszugriffe ... 20
7.1.3 Behandlung multipler Eigenschaften ... 21
7.1.4 Prädikate in Eigenschaftszugriffen ... 21
7.2 Implizite geometrische Eigenschaften ... 23
7.2.1 Alternative 1: Eine Filter-Funktion zur Berechnung der Geometrie ... 24
7.2.2 Alternative 2: Eine XPath-Funktion zur Berechnung der Geometrie ... 24
7.2.3 Alternative 3: Das Einrichten der Geometrie durch Erweiterung des OKSTRA® ... 25
7.3 Umgang mit der OKSTRA®-Historisierung ... 25
7.3.1 Die Historisierung im OKSTRA® ... 26
7.3.2 Alternative 1: Einfache Historienunterstützung auf Basis des OKSTRA® ... 26
7.3.3 Alternative 2: Historienunterstützung mit Komfort ... 27
7.4 Transaktionen und Locks ... 28
7.4.1 Der Leistungsumfang der WFS-Schnittstelle ... 28
7.4.2 „Optimistisches" Locking und der WFS ... 29
7.4.3 Operationsübergreifende Transaktionen ... 31
7.4.4 Verteilte Transaktionen ... 31
7.5 Normalisierung des OKSTRA® ... 32
7.6 OKSTRA®-Profile ... 33
7.7 Vordefinierte Standardabfragen ... 34

7.8 Protokolle der Datenbereitstellungsschnittstelle 34
7.9 Zusätzliche Anforderungen 35

8 OKSTRA®-WFS lesend (normativ) 37
8.1 Profil des Web Feature Service 37
8.1.1 Operation: GetCapabilities 38
8.1.2 Operation: DescribeFeatureType 40
8.1.3 Standard-Parameter 40
8.1.4 Query-Parameter 41
8.1.5 Operation: GetPropertyValue 42
8.1.6 Operation: GetFeature 42
8.1.7 Vordefinierte Abfragen 42
8.1.8 Koordinatenreferenzsysteme 43
8.1.9 Identifier, gml:id und xlink:href Verweise 43
8.1.10 Ausnahmebehandlung 44
8.1.11 Protokolle 46
8.2 Profil des Filter Encoding 46
8.2.1 Property-Zugriffe 46
8.2.2 Geometrische Prädikate 47
8.2.3 Skalare Vergleiche 47
8.2.4 Logik 47
8.2.5 Ausdrücke, Literale, Funktionen 47
8.2.6 Identifier-Filter 47
8.3 Einschränkungen im OKSTRA®-Schema 48
8.3.1 Schlüsseltabellen 48
8.3.2 Strecken- und Bereichsobjekte 48
8.4 Allgemeine zusätzliche Forderungen 48
8.4.1 Performanz 48
8.4.2 Größenbeschränkungen 48

9 OKSTRA®-WFS transaktional (normativ) 49
9.1 Profil des Web Feature Service 49
9.1.1 Operation: GetCapabilities 49
9.1.2 Operation: DescribeFeatureType 50
9.1.3 Operation: GetPropertyValue 50
9.1.4 Operation: GetFeature 50
9.1.5 Vordefinierte Abfragen 50
9.1.6 Operation: GetFeatureWithLock 50
9.1.7 Operation: LockFeature 51
9.1.8 Operation: Transaction 52
9.1.9 Ausnahmebehandlung 56
9.2 Profil des Filter Encoding 56
9.3 Einschränkungen im OKSTRA®-Schema 56
9.4 Allgemeine zusätzliche Forderungen 56
9.4.1 Performanz 56
9.4.2 Größenbeschränkungen 57

10 Authentifizierung und Autorisierung 59
10.1 Authentifizierung 59
10.1.1 http-basierte Authentifizierung 59
10.1.2 Authentifizierung mit SOAP 60
10.1.3 Anforderungen (normativ) 62
10.2 Autorisierung 63
10.2.1 Begriffsbestimmungen 63
10.2.2 Regeltypen für Berechtigungen 65

10.2.3 Anforderungen (normativ) 65
10.2.4 Architektur 66

11 Metadaten 67

1 Zweck des Dokuments (informativ)

1.1 Leserkreis

Das Dokument richtet sich an diejenigen Personen und Institutionen, die mit der Implementierung OKSTRA®-konformer Web-Services und deren Nutzung durch Client-Software befasst sind.

1.2 Kernaussagen des Inhalts

Es wird eine Datenbereitstellungsschnittstelle für OKSTRA®-Daten auf der Basis der Spezifikation des Web Feature Service (WFS) des Open Geospatial Consortiums (OGC) definiert. Die Definition erfolgt in zwei Stufen:

- OKSTRA-WFS lesend
- OKSTRA-WFS transaktional

Die zweite Stufe (OKSTRA-WFS transaktional) beinhaltet die erste und sieht auch den schreibenden Zugriff auf OKSTRA®-Inhalte vor.

Die folgenden Kapitel sind zum Teil als *informativ* und zum Teil als *normativ* gekennzeichnet. Nur die normativen Teile sind Bestandteil der Definition. Die informativen Aussagen dienen lediglich zur Veranschaulichung und Begründung des Vorgehens.

1.3 Änderungen gegenüber der Version 1.0.2 des Dokuments

Die wesentlichen Änderungen betreffen:

- Anpassung an die aktuellen Versionen der Standards zu Web Feature Services und Filter Encoding
- Verwendung von OKSTRA®-Profilen zur Definition des Umfanges des bereitgestellten Schemas
- Vorgaben für authentifizierende OKSTRA-WFS
- Empfehlungen für Autorisierungsmechanismen

Daneben wurden kleinere sachliche und formale Korrekturen vorgenommen, sowie an einigen Stellen zusätzliche Klarstellungen eingefügt.

2 Gegenstand (normativ)

Dieses Dokument definiert eine Datenbereitstellungsschnittstelle für OKSTRA®-Daten auf der Basis der Spezifikation des Web Feature Service (WFS) des Open Geospatial Consortiums (OGC).

Die Schnittstelle besteht aus folgenden Teilen

1. OKSTRA-WFS lesend (verpflichtend, Abschnitt 8.1)
2. Filter Encoding (verpflichtend, Abschnitt 8.2)
3. OKSTRA-WFS transaktional (optional, Abschnitt 9.1)
4. Authentifizierung (optional, Abschnitt 10.1)

Die zweite Stufe (OKSTRA-WFS transaktional) beinhaltet die erste und sieht auch den schreibenden Zugriff auf OKSTRA®-Inhalte vor.

Die Definition ist Teil einer Gesamtspezifikation[1] zu „OKSTRA®-konformen Web-Services", also von Web-Services, welche generell OKSTRA®-Daten bereitstellen und verarbeiten. Neben der Datenbereitstellung gehören dazu auch höherwertige Schnittstellenfunktionen, welche über die reine Bereitstellung hinausgehend allgemein verwendbare „Geschäftslogik" des Straßen- und Verkehrsbereichs umfassen. Höherwertige Schnittstellenfunktionen sind in der vorliegenden Spezifikation nicht enthalten, es werden jedoch verbindliche Vorgaben für Authentifizierungsprotokolle

Es werden Empfehlungen für eine Autorisierungsarchitektur ausgesprochen, diese haben jedoch nicht das Niveau einer ausgearbeiteten Schnittstellendefinition, sondern das einer konzeptionellen Grundlage für eine solche Definition.

Der Zweck der Datenbereitstellungsschnittstelle für OKSTRA®-Daten ist die Kapselung der Verschiedenheiten der existierenden und zukünftigen Speicherungs- und Verarbeitungssysteme für Straßen- und Verkehrsdaten (Straßendatenbanken). Der Zugriff auf diese Systeme erfolgt über die vereinbarte Web-Schnittstelle auf der semantischen Basis des OKSTRA®-XML-Schemas.

Ziel der Definition ist auch Kompatibilität zu Geodateninfrastrukturen. Wenn auch nicht zu erwarten ist, dass das OKSTRA®-Schema als rein deutscher Standard die Basis europaweiter Schemafestlegungen, z.B. im Rahmen von INSPIRE werden wird, so wird es auf der Basis der dort auch eingesetzten Technologie für die Datenbereitstellung (also GML und WFS) leichter fallen, die benötigten Dienste für die Wandlung in die harmonisierten INSPIRE-Datenstrukturen bereitzustellen.

Der OKSTRA-WFS ist weitgehend als ein Profil – d.h. als spezialisierende Untermenge – der OGC WFS-Spezifikation (Version 2.0) definiert. D.h. ein OKSTRA-WFS *ist* (weitgehend) ein OGC WFS. Es gibt eine geringfügige Ausnahme von dieser Regel, bei der die WFS-Spezifikation erweitert und verändert wurde. Sie betrifft die Zulassung eines „optimistischen" Locking-Schemas.

[1] Dies beschreibt einen Planungsstand. Zur Zeit der Schriftlegung dieses Dokuments existiert diese Gesamtspezifikation noch nicht.

3 Regelungen zur Konformität (normativ)

Dieses Kapitel beschreibt, wie für eine Instanz der OKSTRA®-Datenbereitstellungsschnittstelle (OKSTRA-WFS lesend oder OKSTRA-WFS transaktional) die Konformität mit den Definitionen in diesem Dokument und den normativen Bezügen hergestellt wird.

Es wurden bisher noch keine spezifischen Konformitätsprüfungen für die beiden Ebenen

- OKSTRA-WFS lesend und
- OKSTRA-WFS transaktional

entwickelt. Diese müssen aus der Erfahrung mit der vorliegenden Spezifikation in der Praxis entstehen und in die Spezifikation integriert werden. Eine Möglichkeit ist z.B. die Einrichtung einer automatischen Prozedur für eine Konformitätsprüfung. Eine minimale Konformität mit der vorliegenden Spezifikation erfordert die Beachtung der in den folgenden Abschnitten behandelten Punkte.

- Ein *OKSTRA-WFS lesend* muss (außer es ist in der vorliegenden Spezifikation anders definiert) konform sein zu einem OGC Basic WFS Version 2.0, wie er in der normativen Referenz 4.2, ergänzt durch Referenz 4.1 beschrieben ist. Er muss dessen gesamte obligatorische Funktionalität aufweisen. Unter der obligatorischen Funktionalität des OGC Basic WFS wird hier die Funktionalität verstanden,
    - welche im Standard durch entsprechende Verbformen wie „shall" oder „must", etc. ausdrücklich als solche bezeichnet wird oder
    - für die keine Möglichkeit vorliegt, ihr Fehlen im Capabilities-Response-Dokument zu beschreiben.
- Ein *OKSTRA-WFS lesend* muss alle zutreffenden verbindlichen Regelungen der normativen Kapitel der vorliegenden Spezifikation (außer Kapitel 9) erfüllen, insbesondere alle Regelungen des Kapitels 8. Verbindliche Regelungen werden durch die Verbformen „muss" oder „soll" ausgedrückt. Konjunktivische Formen wie „sollte" dagegen drücken Empfehlungen aus.
- Ein *OKSTRA-WFS transaktional* muss (außer es ist in der vorliegenden Spezifikation anders definiert) konform sein zu einem OGC Transactional WFS oder Locking WFS Version 2.0, wie er in der normativen Referenz 4.2 beschrieben ist. Er muss dessen gesamte obligatorische Funktionalität aufweisen. Die obligatorische Funktionalität des OGC Transactional oder Locking WFS ist wie oben beim OGC Basic WFS zu verstehen.
- Ein OKSTRA-WFS transaktional, der konform zu einem Transactional WFS, aber nicht zu einem Locking WFS ist, muss bei der Transaction-Operation automatische Locks unterstützen.
- Ein *OKSTRA-WFS transaktional* muss zusätzlich konform sein zu einem *OKSTRA-WFS lesend*.
- Ein *OKSTRA-WFS transaktional* muss alle zutreffenden verbindlichen Regelungen der normativen Kapitel der vorliegenden Spezifikation erfüllen, insbesondere alle Regelungen der Kapitel 8 und 9. Verbindliche Regelungen werden durch die Verbformen „muss" oder „soll" ausgedrückt. Konjunktivische Formen wie „sollte" dagegen drücken Empfehlungen aus.
- Abweichend von der Regelung in der Spezifikation des OGC Locking WFS Version 2.0 darf ein *OKSTRA-WFS transaktional* „optimistisches" Locking implementieren. Wenn davon Gebrauch gemacht wird, sind 8.1.1.3, 9.1.6 und 9.1.7 zu beachten.

4 Normative Bezüge (normativ)

Die *Datenbereitstellungs-Schnittstelle für OKSTRA®-Daten auf der Basis des OGC Web Feature Service*, kurz *OKSTRA®-WFS*, beruht auf den im Folgenden aufgeführten Standards.

Soweit die hier gelisteten Standardisierungs-Dokumente mit einem Veröffentlichungsdatum oder einer Version versehen sind, so gelten die im vorliegenden Dokument definierten Spezifikationen nur für die so bezeichnete Ausgabe des jeweiligen Standards und nicht für spätere Fortschreibungen. Nutzer der vorliegenden Spezifikation für den *OKSTRA®-WFS* werden jedoch dazu ermutigt, zu untersuchen, inwieweit die neuesten Ausgaben der bezeichneten Standards anwendbar sind, ohne dass die vorliegende Spezifikation Änderungen erfahren müsste.

Soweit die gelisteten Standardisierungs-Dokumente ohne Version oder zeitliche Festlegung existieren, gelten immer die neuesten Versionen.

4.1 Filter Encoding

OpenGIS® Filter Encoding 2.0 Encoding Standard, Open Geospatial Consortium (OGC), 2010-11-22, Ref.No OGC 09-026r1

4.2 Web Feature Service

OpenGIS® Web Feature Service 2.0 Interface Standard, Open Geospatial Consortium (OGC), 2010-11-02, Ref.No OGC 09-025r1

4.3 OKSTRA®

4.3.1 OKSTRA®-Datenmodell

Objektkatalog für das Straßen- und Verkehrswesen, Bundesanstalt für Straßenwesen / OKSTRA®-Pflegestelle, http://www.okstra.de. Aufgrund der Festschreibung des OGC-WFS 2.0 (siehe 4.2) sind alle OKSTRA®-Versionen verwendbar, für die ein GML 3.2-Applikationsschema existiert. Dies sind die OKSTRA®-Versionen ab 2.016 aufwärts.

4.3.2 OKSTRA®-Profile

Festlegung von inhaltlichen Ausschnitten (Profilen) des OKSTRA®-Datenmodells, OKSTRA®-Pflegestelle, 2014-04-11, http://www.okstra.de, Dokument T0009.pdf

4.3.3 Koordinatenreferenzsysteme

Vorschlag zur Neuordnung der Angabe von Koordinatenreferenzsystemen (zu Änderungsantrag A0115), OKSTRA®-Pflegestelle, 2014-10-28, http://www.okstra.de, Dokument N0155.pdf

4.4 Authentifizierung

HTTP Authentication, RFC 2617, http://tools.ietf.org/html/rfc2617

Web Services Security (WSS), OASIS Open 2006-02-01, http://docs.oasis-open.ürg/wss/v1.1/

Teildokumente:

- SOAP Message Security 1.1 (WS-Security 2004), inklusive Korrekturen

- UsernameToken Profile 1.1
- X.509 Certificate Token Profile 1.1
- Kerberos Token Profile 1.1
- SAML Token Profile 1.1

5 Glossar (normativ)

Dieses Kapitel definiert Begriffe, die in dieser Spezifikation verwendet werden.

Capabilities

Ein Client erfragt ein Capabilities-Dokument von einem Web-Service über eine spezielle GetCapabilities-Operation. Ein solches Dokument enthält Metadaten, die den Service beschreiben und spezifische Charakteristika des betroffenen Service.

Client, Web-Client

Softwarekomponente, welche einen Web-Service benutzt.

Feature

Ein Feature ist im Zusammenhang mit dem OKSTRA-WFS ein OKSTRA®-Objekt, bzw. im Falle von Historisierung eine OKSTRA®-Objektversion, d.h. ein Objekt mit einem bestimmten Lebenszeitintervall. Ein Feature ist eine Ausprägung eines FeatureTypes.

FeatureType

Ein FeatureType definiert im vorliegenden Zusammenhang die gemeinsame Struktur von OKSTRA®-Objekten. Alle Features besitzen einen FeatureType.

Koordinatenreferenzsystem

Eine Definition bezüglich der Interpretation von Koordinaten (Vektoren von Zahlen) als Positionen in der Welt.

Operation

Angabe einer Aktion oder einer Abfrage an einen Service, welcher dieser ausführen soll.

Register

Ein Register ist der (Metadaten-)Inhalt einer Registry. Oft wird das Wort verwendet, um einen abgegrenzten Teil dieses Inhalts zu beschreiben.

Registry

Eine Registry ist eine Komponente (ein Web-Service), welche die Daten eines Registers zur Verfügung stellt.

Request

Ein Client ruft eine Operation eines Web-Service auf. Das übermittelte Datenpaket wird als Request bezeichnet.

Response

Das Resultat einer Operation. Der Service gibt den Response an den Client zurück.

Service, Web-Service

Eine Komponente, die über das Netz erreichbar ist und eine festgelegte Schnittstelle von Operationen bereitstellt, die von Clients aufgerufen werden können.

6 Überblick über OGC WFS (informativ)

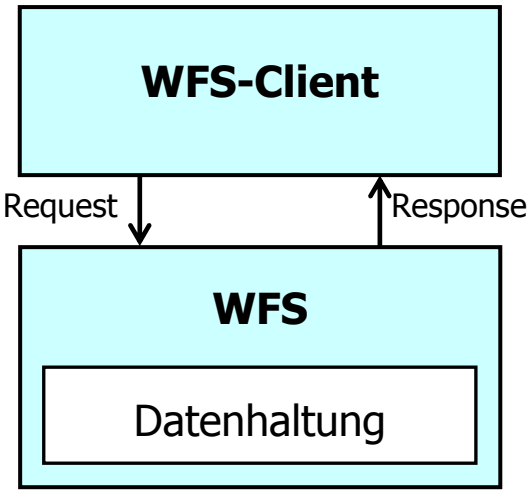
Dieses Kapitel gibt einen kurzen Überblick über OGC WFS 2.0 und Filter Encoding 2.0.

Der OGC Web Feature Service (WFS) stellt eine Schnittstelle zum Zugriff und für Veränderungsoperationen auf Daten mit Raumbezug zur Verfügung. Der Zugriff erfolgt wie bei allen Web-Services über http, das Protokoll des World-Wide-Web.

Die Schnittstelle des WFS dient zur Beschreibung, Abfrage, Erzeugung, Fortschreibung und Löschung von Daten, d.h. es handelt sich um das Äquivalent zu einer Datenbankschnittstelle (vergleichbar zu SQL, aber nicht ganz so leistungsfähig) für den Betrieb als Web-Service. Die Strukturierung der Daten erfolgt dabei gemäß einem Applikationsschema in OGC Geography Markup Language (GML), einer XML Sprache für räumliche Daten.

Für den OKSTRA® liegt eine solche Strukturierung in Form eines GML-Applikationsschemas vor, es nennt sich OKSTRA® XML. OKSTRA®-Daten sind daher für die Abfrage und Pflege durch einen Web Feature Server (eine Software, die einen Web Feature Service realisiert) geeignet. Die OKSTRA®-Objekte werden in diesem Zusammenhang als „Features" bezeichnet.

Sowohl die Abfragesprache, welche durch den OGC Filter Encoding Standard gegeben ist, als auch die zurückgebrachten Abfrageergebnisse (die ermittelte Menge von „Features") beziehen sich auf die Strukturierung der Daten in GML bzw. sind in dieser Form strukturiert. Dies ist unabhängig davon, wie die konkrete Haltung der Daten „hinter" oder „unter" oder „im" WFS beschaffen ist. Das bedeutet, der WFS kapselt und abstrahiert die tatsächliche Natur dieses Speichers und liefert nach außen nur eine Sicht auf die Daten in Form des verwendeten GML-Applikationsschemas, also OKSTRA® XML.



Abfragen (Queries) folgen dem Filter Encoding Standard. Er erlaubt es, wie in anderen Abfragesprachen, z.B. SQL, Datenelemente anzusprechen und durch Vergleiche und logische Kombinationen zu selektieren. Eine Besonderheit beim Filter Encoding sind die Räumlichen Operatoren, welche Bedingungen auf geometrischen Eigenschaften beschreiben. Sie können zusammen mit „normalen" Vergleichen eingesetzt werden, so dass es z.B. möglich wird, bestimmte Objekte in der Nähe von anderen Objekten zu ermitteln.

Der Zugriff auf die Datenelemente erfolgt beim Filter Encoding natürlich in Bezug auf das GML-Applikationsschema, d.h. durch Bezugnahme auf OKSTRA® XML. Infolge der starken Strukturierung der OKSTRA®-Daten ist es dabei wesentlich, dass assoziative Verknüpfungen der OKSTRA®-Objekte einfließen können.

Ein Web Feature Service umfasst die folgenden Operationen:

Web Feature Service	
Operation	Beschreibung
GetCapabilities	Die Fähigkeiten eines WFS werden in einem umfangreichen XML-Dokument beschrieben.
DescribeFeatureType	Durch diese Operation wird das GML-Schema für die angegebenen FeatureTypes abgefragt. Durch das Schema weiß der Client, welche Eigenschaften ein Feature hat.
GetPropertyValue	Diese Operation erfragt den Wert einer einzelnen Eigenschaft eines Features mit Hilfe eines Filter-Ausdrucks.
GetFeature	Die Operation erfragt die Inhalte eines oder mehrerer Features. Die Auswahl der Features erfolgt anhand des FeatureTypes und (optional) eines Filter-Ausdrucks, der es gestattet, Features nach inhaltlichen Kriterien zu selektieren. Die Features werden ausgedrückt in GML (Geography Markup Language) zurückgebracht.
LockFeature	Durch diese Operation werden „Locks" auf durch Filter selektierte Features genommen. Features, auf denen ein Lock besteht, können nicht durch andere Clients verändert werden. Es können auch keine weiteren Locks für diese genommen werden.
GetFeatureWithLock	Wie GetFeature. Die Operation nimmt aber zusätzlich ein „Lock" auf den ausgewählten Features, das sicherstellt, dass diese bis zu einer nachfolgenden Transaction-Operation, die sich auf das Lock bezieht, nicht durch andere Clients verändert werden können.
ListStoredQueries	Gibt eine Liste vordefinierter, im WFS gespeicherter Abfragen zurück. Die Liste enthält wenigstens ein Element, da die vordefinierte Abfrage GetFeatureById verpflichtend eingebaut sein muss.
DescribeStoredQueries	Gibt eine oder mehrere Beschreibungen vordefinierter, im WFS gespeicherter Abfragen zurück.
Transaction	In einer Transaction können beliebig viele Features erzeugt, verändert oder gelöscht werden. Eine Transaction wirkt als Einheit, d.h. ganz oder gar nicht. Sollen Features mit „Locks" an einer Transaction teilnehmen, so muss die zugehörige „Lock-Id" angegeben werden.

Die einfachste Ausprägung eines Web Feature Server heißt „Simple WFS" und unterstützt nur die Operationen „GetFeature mit Stored Query (vordefinierte Abfrage)", „ListStoredQueries", „DescribeStoredQueries", „DescribeFeatureType" und „GetCapabilities". „Basic WFS" unterstützen zusätzlich „GetFeature" mit freien Abfragen sowie „GetPropertyValue". Beide Arten können also nur lesen. Ein WFS, der auch schreiben kann und deshalb die Operation „Transaction" unterstützt, heißt „Transactional WFS" sowie „Locking WFS", wenn er zusätzlich explizite Operationen zum Sperren von Features unterstützt.

7 Entscheidungsgrundlagen (teilweise normativ)

Aufgrund der Struktur des OKSTRA® und der bekannten Anforderungen ergeben sich einige Problemfelder, welche an dieser Stelle analytisch behandelt werden. Das Kapitel leitet aus der Analyse der behandelten Problemfelder verbindliche (normative) Anforderungen an die Funktionalität eines OKSTRA-WFS ab, diese führen zu den in den beiden folgenden normativen Kapiteln getroffenen Festlegungen.

Hinweis zur Terminologie

Bezüglich des Begriffs „Objekt" gibt es im Zusammenhang mit der OKSTRA®-Historisierung eine gewisse sprachliche Unschärfe und eine Abweichung von der normalen Verwendung im IT-Bereich.

Unter einem „Objekt" wird beim OKSTRA® die gesamte sich zeitlich in ihren Eigenschaften verändernde Einheit verstanden (etwa eine bestimmte Straße während ihres Bestehens), während eine der Ausprägungen zu einer Zeit (oder in einem Zeitintervall) als „Version" bezeichnet wird.

Nun ist es aber so, dass die eigentlichen im Schema modellierten Einheiten eben die Versionen sind – die Objekte im obigen Sinne kommen darin gar nicht vor. Dies ist im Gegensatz zum normalen Gebrauch bei objektorientierter Modellierung, wo die Instanzen von Objektarten eben Objekte genannt werden und nicht Versionen.

Um verständlich zu bleiben, werden wir im Folgenden unter „Objekt" die normale Bedeutung im IT-Bereich verstehen, also die Ausprägung (die Instanz) einer Objektart. Nur wenn wir über Historisierung reden, wenden wir die genauen Begriffe an.

7.1 Anforderungen an Filterausdrücke

Dieser Abschnitt analysiert die Anforderungen, die sich durch das OKSTRA®-Schema bezüglich der WFS-Filtersprache ergeben.

7.1.1 Die Filtersprache

Die von der WFS-Spezifikation vorgesehene Filtersprache, dargelegt in der Filter Encoding Spezifikation mit einigen Erweiterungen durch die WFS-Spezifikation, reicht im Großen und Ganzen aus, um die Anforderungen an wirkungsvolle Queries gegen einen OKSTRA®-Bestand zu befriedigen.

Vorhanden sind im aktuellen Filter Encoding Standard:

- Zugriffe auf Eigenschaften der Objekte,
- Konstanten,
- Funktionsaufrufe,
- Vergleiche (=, <>, <, <=, >, >=), einschließlich Intervallabfrage (Between), Mustervergleich (Like) und NULL-Vergleich (IsNull),
- Räumliche Operatoren (BBOX, Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains, DWithin, Beyond),
- Zeitliche Operatoren (After, Before, Begins, BegunBy, TContains, During, TEquals, TOverlaps, Meets, OverlappedBy, MetBy, Ends, EndedBy, AnyInteracts)
- Und, Oder, Nicht

- Spezielle Abfragen nach Objektidentifikatoren

Die Funktionsaufrufe sind im Standard nicht weiter ausspezifiziert und können deshalb für die nötigen Erweiterungen ausgestaltet werden. Hierauf wird bei der Betrachtung der einzelnen Problemfelder eingegangen werden.

Die in der Version 1.1 des Filter Encoding Standards noch enthaltenen Grundrechenarten sind in der Version 2.0 weggefallen und müssen durch entsprechende Funktionen emuliert werden.

Erfahrungsgemäß werden die meisten Elemente der Filtersprache tatsächlich benötigt. Allenfalls bei den räumlichen Operatoren können evtl. einige seltener benötigte entfallen. Benötigt werden auf alle Fälle BBOX, Disjoint, Within, Intersects, Contains, und DWithin, da sie verschiedene Varianten des Enthaltenseins, bzw. Nichtenthaltenseins in Bezug auf Geometrievorgaben ausdrücken. Zusätzlich ist Overlaps eine häufig eingesetzte Relation und sollte vorhanden sein.

Auch die Zeitvergleichsoperatoren sind nicht zwingend in diesem OKSTRA-WFS-Profil vorgesehen. Falls sie teilweise oder komplett implementiert werden, sollen sie auf alle Eigenschaften der OKSTRA®-Objekte wirken können, die Zeitpunkte oder Zeiträume beschreiben. In diesem Fall ist mindestens der Operator During zu implementieren.

Anforderung

Die Filtersprache soll mit Ausnahme der Zeit- und Geometrie-Operatoren vollumfänglich unterstützt werden. Mindestens folgende Geometrie-Operatoren sollen vorhanden sein: BBOX, Disjoint, Within, Intersects, Contains, DWithin. Es sollen Funktionen für die Grundrechenarten Add, Sub, Mul und Div mit 2..* Operanden unterstützt werden.

7.1.2 Eigenschaftszugriffe

Wirkliche Erweiterungen gegenüber der Spezifikation sind im Zusammenhang mit den Zugriffsmöglichkeiten auf die Eigenschaften der Objekte erforderlich.

Der Grund liegt darin, dass das OKSTRA®-Schema, der administrativen Wirklichkeit folgend, in den meisten Bereichen relativ komplex ist. Es gibt eine Vielzahl von Objektarten und relationale Bezüge zwischen diesen, welche zu beachten sind.

Bei der Abfassung von Queries bezüglich einer Objektart (FeatureType), wie es durch die WFS-Operation GetFeature geschieht, besteht deshalb nur in den einfachsten Fällen die Möglichkeit, sich auf die direkten Eigenschaften der erfragten Objektart zu beschränken.

Einfaches Beispiel:

Finde alle Abschnitte, die zu Autobahnen gehören.

Um herauszufinden, zu welcher Straßenklasse ein *Abschnitt* gehört, muss man vom *Abschnitt* ausgehend der Relation *gehoert_zu_Strasse*, in der *Strasse* der Relation *hat_Strassenbezeichnung* und in der *Strassenbezeichnung* der Relation *Strassenklasse* folgen. In der *Strassenklasse* ist die Eigenschaft *Kennung* auszuwerten (für Autobahnen = „A").

Das Query-Konstrukt in der WFS GetFeature-Operation sieht folgendermaßen aus[2]:

```
<wfs:Query typeNames='Abschnitt'>
 <fes:Filter>
  <fes:PropertyIsEqualTo>
   <fes:ValueReference>gehoert_zu_Strasse/Strasse/hat_Strassenbezeichnung/
```

[2] Das Beispiel vernachlässigt die Historisierung. Siehe zu diesem Thema die Erörterung weiter unten.

```
    Strassenbezeichnung/Strassenklasse/Strassenklasse/Kennung</fes:ValueReference>
   <fes:Literal>A</fes:Literal>
  </fes:PropertyIsEqualTo>
 </fes:Filter>
</wfs:Query>
```

Die beim Element <fes:ValueReference> verwendete XPath-Notation ist zwar im Filter-Encoding- und im WFS-Standard vorgesehen, aber es ist in der aktuellen Version 2.0 der Standards nicht explizit spezifiziert, dass damit die Eigenschaften anderer Objektarten angesprochen werden können.

Dies muss jedoch auf alle Fälle gefordert werden, um OKSTRA®-Objekte sinnvoll inhaltlich erfragen zu können. Der AFIS-ALKIS-ATKIS-Standard der deutschen Vermessungsverwaltungen verwendet ebenfalls diese Regelung.

Anforderung

Die XPath-Notation für Eigenschaftszugriffe (PropertyName) soll es gestatten, Relationen zu anderen Objektarten (FeatureTypes) zu verfolgen, auch wenn diese im OKSTRA® XML Schema durch eine Referenz (xlink:href) ausgedrückt sind. Ein solcher Zugriff repräsentiert in seiner Gesamtheit die Menge der so vom Objekt (Feature) aus erreichten Eigenschaftswerte.

Folgeversionen des WFS werden aller Voraussicht nach diese erweiterte Interpretation der XPath-Notation für den Eigenschaftszugriff in einer abgewandelten Syntax unterstützen.

7.1.3 Behandlung multipler Eigenschaften

Unabhängig davon, ob man gestattet, bei Eigenschaftszugriffen Relationen zu anderen Objekten zu verfolgen (aber besonders dann) muss man mit dem Problem umgehen, dass die angesprochenen Eigenschaften multipel vorliegen. Es ist zu klären, wie z.B. ein multipler Wert in einem Vergleich wirkt.

Der Filter-Encoding-Standard sieht hierfür den Parameter matchAction bei den Vergleichsoperatoren vor. Dieser kann die folgenden Werte annehmen:

- All: alle Werte der Eigenschaft müssen den Vergleich erfüllen.
- Any: wenigstens ein Wert der Eigenschaft muss den Vergleich erfüllen.
- One: genau ein Wert der Eigenschaft muss den Vergleich erfüllen.

Zusätzlich fordert der Filter-Encoding-Standard eine minimale XPath-Implementation, die es gestattet, das erste, zweite oder n-te Element einer Menge von Eigenschaften zu selektieren.

Ein Beispiel für den nützlichen Einsatz dieser Positionsprädikate ist z.B. die Selektion eines Teilabschnitts in einer Strecke.

7.1.4 Prädikate in Eigenschaftszugriffen

Für Teilbereiche des OKSTRA® sind noch weitergehende Anforderungen an die Eigenschaftszugriffe zu stellen. Sie benötigen über die oben geforderte Interpretation der XPath-Ausdrücke hinaus die Anwendbarkeit von zusätzlichen Prädikaten in den einzelnen Pfadpositionen.

Ein Beispiel aus dem Bereich Verkehrsdaten:

Es seien Zählstellen (im Beispiel der Objektart *automatische_Dauerzaehlstelle*) zu erfragen, welche für ein bestimmtes *Bezugsjahr* und für die *Fahrzeugart* Pkw einen DTV-Wert von 100000 überschreiten.

Dies kann eigentlich nur durch ein Query folgender Form erfolgen:

```
<wfs:Query typeNames='automatische_Dauerzaehlstelle'>
 <fes:Filter>
  <fes:PropertyIsGreaterThan>
   <fes:ValueReference>zu_DTV/DTV[Bezugsjahr=2005 and Fahrzeugart/Fahrzeugart/Kennung='Pkw']/
    Fahrzeuge_pro_24h</fes:ValueReference>
   <fes:Literal>100000</fes:Literal>
  </fes:PropertyIsGreaterThan>
 </fes:Filter>
</wfs:Query>
```

Der Versuch, dies mit konventionellen Mitteln anders auszudrücken, etwa durch ein AND-Konstrukt, bei dem der Vergleich und das Prädikat verknüpft werden, bewirkt nicht dasselbe.

```
<wfs:Query typeNames='automatische_Dauerzaehlstelle'>
 <fes:Filter>
  <fes:And>
   <fes:PropertyIsEqualTo>
    <fes:ValueReference>zu_DTV/DTV/Bezugsjahr</fes:ValueReference>
    <fes:Literal>2005</fes:Literal>
   </fes:PropertyIsEqualTo>
   <fes:PropertyIsEqualTo>
    <fes:ValueReference>zu_DTV/DTV/Fahrzeugart/Fahrzeugart/Kennung</fes:ValueReference>
    <fes:Literal>Pkw</fes:Literal>
   </fes:PropertyIsEqualTo>
   <fes:PropertyIsGreaterThan>
    <fes:ValueReference>zu_DTV/DTV/Fahrzeuge_pro_24h</fes:ValueReference>
    <fes:Literal>100000</fes:Literal>
   </fes:PropertyIsGreaterThan>
  </fes:And>
 </fes:Filter>
</wfs:Query>
```

Dieser Filterausdruck löst die gegebene Selektionsaufgabe nicht, da die drei Komponenten des *And* voneinander unabhängige Ergebnisse bezüglich der DTV-Objekte liefern[3].

Anforderung

Die XPath-Ausdrücke in Eigenschaftszugriffen (PropertyName) sollen Prädikate in Sinne der XPath-Syntax unterstützen. Die Unterstützung ist erforderlich auf den Pfadpositionen, welche den OKSTRA®-Objekten (Features) entsprechen. Die Ausdruckssyntax soll umfassen: XPath-Eigenschaftszugriffe, Konstanten, Vergleiche, and, or, not().

[3] Über die Einzelkomponenten hat man zunächst drei Ergebnismengen von automatischen Dauerzählstellen:

1. Solche, die DTVs mit dem Bezugsjahr 2005 besitzen.
2. Solche, die DTVs für die Fahrzeugart „PkW" besitzen.
3. Solche, die DTVs mit einem „Fahrzeuge_pro_24h"-Wert von mehr als 100000 besitzen.

Über den AND-Operator gelangen diejenigen automatischen Dauerzählstellen in die Ergebnismenge des gesamten Filters, die in allen Ergebnismengen der einzelnen Komponenten auftreten. Leider ist damit nicht sichergestellt, dass alle geforderten Eigenschaften im selben DTV auftreten – sie können auch auf unterschiedliche DTVs verteilt sein. Insofern liefert diese Query potenziell zu viele Dauerzählstellen zurück.

Das folgende kleine Beispiel erläutert dies nochmal:

Zählstelle	DTV		
	Bezugsjahr	Fahrzeugart	Fahrzeuge_pro_24h
Z1	2005	PKW	150000
Z2	2005	LKW	107000
	2006	PKW	110000

Das erste Query (mit Prädikat) liefert nur Z1 zurück, weil nur im dort zugeordneten DTV alle drei Bedingungen erfüllt sind. Das zweite Query liefert mit dem ersten Vergleich im AND die Zählstellen Z1 und Z2, weil beide DTVs zum Jahr 2005 haben, mit dem zweiten Vergleich wieder Z1 und Z2, weil beide DTVs zu PKWs haben, und mit dem dritten Vergleich wieder Z1 und Z2, weil alle DTVs die Größenbedingung erfüllen. Das logische AND ergibt gibt dann natürlich auch Z1 und Z2.

Prädikate erleichtern im Übrigen die Behandlung multipler Eigenschaftszugriffe und sind unbedingt erforderlich, wenn die Historisierung im weiter unten vorgeschlagenen Sinne gelöst werden soll.

7.2 Implizite geometrische Eigenschaften

Objekte, welche dem Straßennetz zugeordnet sind, erhalten diesen Netzbezug typischerweise durch „Lineare Referenzierung", im OKSTRA® als Stationierung bezeichnet. Stationierte Objekte treten als OKSTRA®-Objektarten auf, welche Bezüge auf die Abschnitte oder Äste besitzen und gleichzeitig angeben, wo entlang der Geometrie des referierten Abschnitts oder Asts sie positioniert sind.

Bei punktförmigen Bezügen ist das einfach und wird über das Straßenpunkt-Konstrukt hergestellt.

Bei linearen Bezügen, gibt es die erste und einfache Variante, die sich nur auf einen Abschnitt oder Ast bezieht. Hier kommt das Teilabschnitt-Konstrukt zum Tragen, das zum Abschnitt oder Ast einfach zwei Stationen angibt. Die zweite Variante bezieht sich auf die lineare Folge zusammenhängender Teilabschnitte, die zu einer sog. Strecke zusammengefasst sind.

Alle diese Konstruktionen sind in OKSTRA® XML definiert und können in Filterausdrücken abgefragt werden.

Durch die Stationierung erhält das stationierte Objekt implizit eine Geometrie, die (bis auf mögliche seitliche Auslenkung) mit Teilen der Netzgeometrie übereinstimmt. Es ist wünschenswert, diese implizit definierte Geometrie bei Queries nutzbar zu machen. Dies soll in diesem Abschnitt diskutiert werden. Um ein Beispiel zu geben: Man will alle Verkehrszeichentraeger-Objekte (das sind Punktobjekte) in einer vorgegebenen Fläche finden.

Implizite Geometrie kommt aber im OKSTRA® nicht nur infolge der Stationierung vor. Z.B. wird durch die Abschnitte und Äste implizit eine Geometrie der Strassenobjekte gebildet. Weitere Beispiele für implizite Geometrie im OKSTRA® finden sich insbesondere im Schema „Entwurf" (Achselement, Deckenbuch etc.).

Es sind hier im Grunde drei Alternativen denkbar, die leider alle ihre Vor- und Nachteile aufweisen.

7.2.1 Alternative 1: Eine Filter-Funktion zur Berechnung der Geometrie

Wie in 7.1.1 bereits angemerkt, besitzt die Filtersprache ein Konstrukt für sog. Functions. Diese sind im Filter Encoding Standard nicht festgelegt, ein WFS-Profil kann jedoch davon Gebrauch machen und spezielle *Functions* definieren und fordern.

Es wäre in diesem Falle eine *Function* zu definieren, welche bei Versorgung mit dem entsprechenden Stationierungskonstrukt (also Straßenpunkt oder Teilabschnitt oder Strecke) und – im Falle der Historisierung – einem zusätzlichen Bezugsdatum die Berechnung der gewünschten Geometrie vornimmt und als Wert der *Function* die betreffende Geometrie zurückbringt.

Die zurückgebrachte Geometrie sollte dann wie eine normale Geometrieeigenschaft in räumlichen Operatoren eingesetzt werden.

Leider ist das bei der gegenwärtigen Definition des Filter Encoding nicht möglich, da bei den räumlichen Operatoren statt eines Ausdrucks nur ein Eigenschaftszugriff (ValueReference) erlaubt ist. Es wäre deshalb erforderlich, das Filter Encoding Schema zu erweitern und – wegen dessen enger Verknüpfung mit dem WFS Schema – auch dieses.

Vom Ansatz sähe das etwa so aus:

```
<okwfs:Query typeNames='Verkehrszeichentraeger'>
 <okfe:Filter>
  <okfe:BBOX>
   <fes:Function name='PunktAusStrassenpunkt'>
    <fes:fes:ValueReference>bei_Strassenpunkt</fes:fes:ValueReference>
    <fes:Literal>20071201</fes:Literal>
   </fes:Function>
   <gml:Envelope>
    <gml:lowerCorner> … </gml:lowerCorner>
    <gml:upperCorner> … </gml:upperCorner>
   </gml:Envelope>
  </okfe:BBOX>
 </okfe:Filter>
</okwfs:Query>
```

Dabei wurde das Namespace-Kürzel „okwfs" für das modifizierte WFS-Schema gewählt und „okfe" für das modifizierte Filter Encoding Schema.

Wegen der erforderlichen Erweiterungen der OGC-Spezifikationen (in daraus abgeleiteten Schemas) sollte man diese Alternative nur verfolgen, wenn solche schwerwiegenden Eingriffe aufgrund anderer Anforderungen nötig werden.

7.2.2 Alternative 2: Eine XPath-Funktion zur Berechnung der Geometrie

Die Schwierigkeit aus Alternative 1 kann man vermeiden, indem man die Berechnung der Geometrie in den XPath-Ausdruck des Eigenschaftszugriffs (PropertyName) verlagert.

Dies könnte etwa so aussehen:

```
<wfs:Query typeNames='Verkehrszeichentraeger'>
 <fes:Filter>
  <fes:BBOX>
   <fes:ValueReference>punktAusPunktobjekt(.,'20071201')</fes:ValueReference>
   <gml:Envelope>
    <gml:lowerCorner> … </gml:lowerCorner>
    <gml:upperCorner> … </gml:upperCorner>
   </gml:Envelope>
  </fes:BBOX>
 </fes:Filter>
</wfs:Query>
```

Der Vorteil der Lösung ist, dass weder das Filter Encoding Schema noch das WFS-Schema erweitert werden muss. Es ist nur eine textliche Verschärfung der Spezifikationen erforderlich. Zudem hat die Lösung den Vorteil, dass sie auch zur Geometriebereitstellung in Funktionen zur Kartenerzeugung eingesetzt werden kann, z.B. in einen Styled Layer Descriptor.

Die Lösung erfordert die geringsten Änderungen an den Basisspezifikationen. Allerdings ist die Realisierung aufwändig, so dass sie nicht verpflichtend gemacht werden sollte.

7.2.3 Alternative 3: Das Einrichten der Geometrie durch Erweiterung des OKSTRA®

Stationierte Objekte könnten durch Erweiterung der entsprechenden Stationierungskonstrukte durch eine optionale Geometrie-Eigenschaft aufgerüstet werden.

Diese Geometrie-Eigenschaft würde im OKSTRA® optional eingerichtet, aber für die Verwendung des OKSTRA® für OKSTRA®-konforme Web-Services würde man fordern, dass diese in den stationierten Objekten enthalten sind. Sie könnte in Queries beliebig eingesetzt werden.

Grundsätzlich wäre dies als berechnete Eigenschaft aufzufassen. D.h. sie wäre bei lesenden Operationen sichtbar, würde aber bei schreibenden Operationen ignoriert.

Der Nachteil dieser Lösung ist die erforderliche OKSTRA®-Erweiterung, was bedeutet, dass die Änderung für alte OKSTRA®-Versionen nicht zur Verfügung steht.

Auf der positiven Seite steht, dass wirklich eine konkrete Geometrie zur Verfügung steht, die auch im OKSTRA® XML Response enthalten ist und z.B. für die Kartierung benutzt werden kann.

Anforderung

Neue Versionen des OKSTRA® sollen berechnete Geometrie für ausgewählte Objektarten unterstützen. Für OKSTRA®-WFS, die ältere Versionen unterstützen müssen, sollen die XPath-Ausdrücke in Eigenschaftszugriffen (PropertyName) optional Funktionen zur Berechnung der Geometrie aus OKSTRA®-Stationierungsangaben enthalten können.

7.3 Umgang mit der OKSTRA®-Historisierung

Da die Historisierung der OKSTRA®-Objekte optional ist, ergibt sich die grundsätzliche Fragestellung, ob die OKSTRA®-konformen Web-Services und insbesondere die Datenbereitstellungs-Schnittstelle über OGC WFS, die Historisierung überhaupt unterstützen soll oder nicht. Hier wurde im Vorfeld aufgrund des Anforderungsprofils eine Entscheidung zugunsten der Unterstützung der Historisierung getroffen.

Wenn Historisierung im Gesamtkonzept der OKSTRA®-konformen Web-Services unterstützt werden soll, so hat man die Wahl, auf welcher Ebene dies erfolgen soll. Dieses Dokument behandelt nur die Datenbereitstellung auf der Basis des OGC WFS – es ist also zu festzulegen, in welcher Form die Historienunterstützung im WFS definiert werden kann.

Die WFS 2.0-Spezifikation zeigt die Möglichkeit auf, WFS mit Versionsunterstützung einzurichten. Dabei soll etwa bei einem UPDATE eine neue Version des in der Datenhaltung vorhandenen Objektes angelegt werden. Es gibt Abfrageoperatoren, die Versionsketten ablaufen können, allerdings sind diese nicht mit anderen Abfrageoperatoren kombinierbar. Insgesamt scheint das Konzept nicht gut zur Abbildung der OKSTRA®-Historisierung zu passen. Die WFS-Versionierung scheint eher den Lebenszyklus der Daten abzubilden, während die OKSTRA®-Historisierung den Lebenszyklus des Realweltobjekts betrifft.

In der Version 1.0.2 dieser Spezifikation wurden bereits zwei alternative Herangehensweisen skizziert.

Die erste Alternative unterstützt die Historisierung rein auf der Basis des OKSTRA®-Schemas, kein besonderer Komfort ist vorgesehen. In diesem Falle müsste eine Komfortunterstützung in einer höheren Schnittstellenschicht erfolgen.

Die zweite Alternative skizziert eine komfortable Historienunterstützung für die Datenbereitstellung, hier gibt es einen breiten Spielraum, der bis zu erheblichen Erweiterungen an der WFS-Schnittstelle führen kann (der damit kein WFS mehr wäre).

Zur Einführung dient eine kurze Zusammenfassung der Elemente der OKSTRA®-Historisierung.

7.3.1 Die Historisierung im OKSTRA®

Die meisten Objekte des OKSTRA® unterliegen optional der Historisierung. D.h. sie (die Versionen) besitzen qua Ableitung von *historisches_Objekt* optionale, tagesscharfe Gültigkeitsgrenzen, *gueltig_von*, *gueltig_bis*.

Neben den Gültigkeitsgrenzen in den historischen Objekten des OKSTRA® wird die Historisierung noch durch eine Reihe von Hilfsobjekten unterstützt. Die Zusammenfassung der verschiedenen historischen Objektversionen, die alle nur Varianten ein und desselben zeitveränderlichen Modellgegenstands (im OKSTRA® „Objekt" genannt) sind, wird durch eine lineare Verkettung der historischen Objektversionen beschrieben. Zeitveränderliche Objekte als solche sind im OKSTRA® nicht direkt repräsentiert.

Ereignis-Objekte können Erzeugung und Löschung von historischen Objektversionen, die zu einer Maßnahme gehören, dokumentieren. Die oft komplexen Übergänge bei Veränderungen in der Stationierung können durch *identisches_Netzteile* oder alternativ über das Netzänderungsprotokoll beschrieben werden.

Die Relationen zwischen den Objekten des OKSTRA® unterliegen nur implizit der Historisierung. Sie tragen selbst keine Gültigkeitsgrenzen, sondern gewinnen diese aus den historischen Objektversionen, welche sie miteinander verknüpfen. D.h. die impliziten Gültigkeitsgrenzen einer Relationsinstanz sind durch den Schnitt der Gültigkeitsintervalle der beiden beteiligten historischen Objekte bestimmt.

7.3.2 Alternative 1: Einfache Historienunterstützung auf Basis des OKSTRA®

Als einfachste Lösung bietet es sich an, die Historisierung des OKSTRA® unmittelbar durch Verwendung der dafür vorgesehenen Objekteigenschaften, also insbesondere der Gültigkeitsgrenzen, *gueltig_von*, *gueltig_bis* umzusetzen.

Bei lesenden Aufrufen muss unter Verwendung dieser Grenzen durch entsprechende Vergleiche dafür gesorgt werden, dass nur die Objekte zum gewünschten Stichtag erfragt werden.

Dies ist leicht möglich, beschreibt das Problem aber nur zum Teil. Denn es ist dabei zu bedenken, dass Relationen zu Objektversionen bestehen können, die nicht für den Stichtag gültig sind. Will man über XPath-Ausdrücke auf die Inhalte solcher relational verknüpfter Objekte zugreifen (siehe 7.1.2), so ist sicherzustellen, dass hier keine stichtags-ungültigen Objektversionen erfasst werden.

Das geht nur mit den in 7.1.4 geforderten Prädikaten.

Das Beispiel aus 7.1.2 muss bei einer Abfrage zum Stichtag folgendermaßen abgefragt werden:

```
<wfs:Query typeNames='Abschnitt'>
 <fes:Filter>
  <fes:And>
   <fes:PropertyIsBetween>
```

```
    <fes:Literal>2005-03-25</fes:Literal>
    <fes:LowerBoundary>
      <fes:ValueReference>gueltig_von</fes:ValueReference>
    </fes:LowerBoundary>
    <fes:UpperBoundary>
      <fes:ValueReference>gueltig_bis</fes:ValueReference>
    </fes:UpperBoundary>
    </fes:PropertyIsBetween>
    <fes:PropertyIsEqualTo>
    <fes:ValueReference>gehoert_zu_Strasse/
      Strasse['2005-03-25' >= gueltig_von and gueltig_bis >= '2005-03-25']/hat_Strassenbezeichnung/
      Strassenbezeichnung/Strassenklasse/Strassenklasse/Kennung</fes:ValueReference>
    <fes:Literal>A</fes:Literal>
  </fes:PropertyIsEqualTo>
 </fes:And>
</fes:Filter>
</wfs:Query>
```

In der Version 1.0.2 dieser Spezifikation wurde vorgeschlagen, diesen Ansatz zu verbessern, indem man das Attribut <wfs:Query featureVersion="xxx"> einsetzt, um Stichtage vorzugeben. Allerdings ist dieses Attribut und der entsprechende URL-Parameter nicht mehr in der WFS 2.0-Spezifikation dokumentiert, wenn er auch nach wie vor im WFS-XML-Schema vorhanden ist. Da unter diesen Umständen nicht mehr sicher ist, dass WFS-Clients, die nicht speziell für die Verwendung mit OKSTRA®-WFS-Diensten geschrieben wurden, dieses Attribut interpretieren können, wird dieser Ansatz hier nicht mehr empfohlen.

Bei schreibenden Aufrufen steht ein solches einfaches Vehikel zur Stichtagssteuerung nicht zur Verfügung. Hier muss der der Client sicherstellen, dass die vom OKSTRA® geforderten Invarianten eingehalten werden. D.h. die Gültigkeitsbereiche der Versionsvorgänger sind durch Update einzuschränken und die neue Objektversion ist entsprechend durch eine Insert-Operation einzufügen. Ereignis-Objekte und identische Netzteile sind zu pflegen.

Zumindest bei den schreibenden Operationen erscheint es lohnend, einen entsprechenden Fach-Service, der dies bewerkstelligen kann, zu definieren.

Obwohl die Handhabung von Historisierung nach den Regeln des OKSTRA® zumindest für das Schreiben mühsam erscheint, so ist es doch möglich und relativ klar definiert. Es handelt sich auch um eine Möglichkeit, die immer zur Verfügung stehen wird (und muss), obwohl in der WFS-Schicht oder darüber bequemere Möglichkeiten eingerichtet werden könnten. In diesem Sinne kann zurzeit eigentlich nur diese Variante ausgewählt werden.

Anforderung

Die Unterstützung der OKSTRA®-Historie erfolgt ohne zusätzliche Erweiterungen der WFS-Schnittstelle. Erforderlich sind die an anderer Stelle geforderten „relations-übergreifenden" XPath-Ausdrücke und Prädikate an den Objekten.

Die Unterstützung weitergehender Komfort-Operationen erfolgt oberhalb der Datenbereitstellungs-schicht.

7.3.3 Alternative 2: Historienunterstützung mit Komfort

Eine komfortable Form der Historienunterstützung würde

- es bei lesenden Zugriffen erlauben, den Datenbestand im WFS auf einfache Weise auf einen bestimmten Zeitpunkt oder ein Zeitintervall einzuschränken, und

- bei schreibenden Zugriffen für ein gegebenes Änderungsdatum oder Änderungsintervall, die automatische Aufteilung von historischen Objekten vornehmen. Hierbei sollten bestehende Ereignis-Objekte automatisch fortgeschrieben werden.

Tatsächlich ist eine solche komfortable Historienunterstützung sehr schwer durch ein Profil des OGC WFS zu erreichen.

Das Lesen kann wie in der Alternative 1 erreicht werden. Dies ließe sich evtl. auch für Zeiträume erweitern.

Beim Schreiben liegen die Anforderungen an die WFS-Schnittstelle noch viel höher.

Neue Objekte ohne Bezug auf ältere Objekte werden durch Insert erzeugt.

Durch eine Update-Operation mit einem übergebenen Bezugsdatum (Erweiterung der Schnittstelle) wird automatisch die Filterung der Objekte unterstützt und die Aufteilung der betroffenen Objekte bewirkt, falls das erforderlich ist. Die neuen Objekte müssten durch ein *UpdateResult* analog zum bestehenden *InsertResult* nachgewiesen werden.

Durch eine Delete-Operation mit einem übergebenen Bezugsdatum (Erweiterung der Schnittstelle) würde auch hier automatisch die Filterung der Objekte unterstützt und die qualifizierenden Objekte werden beendet.

Es ist zu bedenken, dass neben der beschriebenen Komfortschnittstelle auch noch die normale in 7.3.2 beschriebene Schnittstelle existieren muss, um Korrekturen an Objekten ohne Historienbildung zu gestatten.

7.4 Transaktionen und Locks

Dieser Abschnitt diskutiert die Fragestellungen zur Profilbildung des WFS bezüglich der gesicherten Durchführung schreibender Operationen. In einem Nutzungsumfeld, in dem die Datenbereitstellung konkurrierend lesend und schreibend durch mehrere Clienten benutzt wird, muss sichergestellt werden, dass von keiner Partei inkonsistente Zwischenzustände gesehen werden können oder sich durch Fehlersituationen permanent manifestieren können. Gewöhnlich wird dies durch die Stichworte Serialisierung und *atomicity, consistency, isolation, durability* – kurz ACID adressiert.

In der Datenbanktechnologie wird dies üblicherweise durch die Verwendung von Transaktionen und Locks (Sperren) gewährleistet. Der WFS-transactional unterstützt diese Konzepte – zu einem gewissen Umfang. Es ist in diesem Kapitel zu analysieren, wie weit man mit dem Standardverhalten kommt und was man verschärfend in der Schnittstelle bereitstellen kann.

7.4.1 Der Leistungsumfang der WFS-Schnittstelle

Die WFS-Spezifikation besitzt folgende Hilfsmittel zur Serialisierung konkurrierender Fortschreibungsoperationen, nämlich:

- Die atomare (also ganz-oder-gar-nicht) Ausführung zusammengesetzter Insert/Update/Replace/Delete-Operationen. (Deshalb heißt diese WFS-Operation auch „Transaction".)
- Die implizite oder explizite Isolation der Zugriffe auf einzelne Features durch „Locks".

Mit diesen Hilfsmitteln ist die gesicherte Fortführung von Datenbeständen im Großen und Ganzen möglich.

Bei Verwendung expliziter Isolation muss die Programmierung eines transaktionssicher ändernden Dienstes so erfolgen, dass zuerst alle beteiligten Features unter Vergabe eines (!) Locks (also durch ein *GetFeatureWithLock* oder ein *LockFeature*) geholt und gesperrt werden müssen. Sobald dies erfolgreich geschehen ist, können die vorgenommenen Veränderungen im Prinzip durch eine oder mehrere *Transaction*-Operationen unter Bezugnahme auf dieses Lock zurückgeschrieben werden.

Mehrere *Transaction*-Operationen dürfen es aber nur sein, wenn bereits nach der ersten solchen ein konsistenter Zustand erreicht ist. Ansonsten muss der gesamte Schreibvorgang in einer *Transaction* zusammengefasst werden.

Bei Verwendung impliziter Isolation übernimmt die Transaction-Operation selbst das Locking (*AutomaticDataLocking*). Dies wird durch den AutomaticDataLocking-Constraint in den Operations Metadata im Capabilities-Document angezeigt. Unterstützt ein WFS implizite Isolation, ist die Implementation der Operationen LockFeature und GetFeatureWithLock optional. Im Falle automatischen Lockings wird in der Transaction-Operation keine Lock-Id angegeben. Die Verwendung impliziter Isolation ist deutlich näher am Transaktionsverhalten von klassischen Datenbanksystemen, da zwischen Sperrung und Veränderung der Daten kein Netzwerkverkehr stattfinden muss. Es handelt sich aber auch hierbei nicht um operationsübergreifende Transaktionen, da nach jeder Transaction auch hier ein konsistenter Zustand hergestellt sein muss.

Die durch den WFS-Standard definierten Locks werden teilweise kritisch bewertet, da sie möglicherweise durch zu restriktives Verhalten die Nutzung der Datenhaltung zu stark einschränken. Es soll daher auch möglich sein, eine „optimistische" Locking-Strategie zu implementieren. Bei dieser wird beim Lesen zum Zwecke des späteren Schreibens die Zeit an den Objekten vermerkt und dem Client mitgeteilt. Beim Schreiben wird geprüft, ob dieser Zeitvermerk unverändert geblieben ist – ansonsten wird der Schreibvorgang zurückgewiesen. Siehe hierzu auch den folgenden Abschnitt 7.4.2.

Eine Entscheidung fällt angesichts der Beschränkungen des vorhandenen Konzepts nicht leicht. Da allerdings alle weitergehenden Definitionen schwerwiegende Erweiterungen des WFS-Standards darstellen, wird empfohlen, erstmal bei der Standardfunktionalität (erweitert um „optimistische" Locks) zu bleiben.

Anforderung

Die Datenbereitstellungsschnittstelle über WFS soll in ihrer schreibenden Ausbaustufe *Transactions* unterstützen. Falls keine expliziten *Locks* implementiert werden, muss Automatic Data Locking unterstützt werden, andernfalls sind die Operationen LockFeature und GetFeatureWithLock zusätzlich zu unterstützen. Es soll dabei auch möglich sein, eine „optimistische" Locking-Strategie umzusetzen.

Ein weitergehendes Profil – wenn es denn gebraucht wird – sollte auf alle Fälle als dritte Ausbaustufe definiert werden, etwa „OKSTRA – Datenbereitstellung, verteilt-transaktional".

7.4.2 „Optimistisches" Locking und der WFS

Die WFS-Spezifikation verlangt, dass zu einer Zeit höchstens ein Lock für ein Feature existieren darf und das Veränderungen an Features, für die ein Lock gesetzt wurde, nur durch die Transactions möglich sind, welche die LockId des Locks kennen. Für einen Clientprozess ergibt sich, dass Features, welche er durch die entsprechenden Operationen (LockFeature bzw. GetFeatureWithLock) mit Locks belegt hat, nur von ihm (mittels der Operation Transaction) verändert werden können, denn kein zweiter Clientprozess kann dieselben Features mit Locks belegen oder verändern. Jeder

Versuch, ein zweites Lock auf einem Feature zu nehmen oder ein „gelocktes" Feature illegal zu verändern, würde nach der WFS-Spezifikation abgewiesen.

Es handelt sich bei den Locks der WFS-Spezifikation also um wirkliche Sperren. Der WFS oder die Datenhaltung müssen sich merken, welche Features in diesem Sinne gesperrt sind und diese Sperren während der definierten Lebenszeit der Locks aufrechterhalten. Diese aktive Form des Lockings, die davon ausgeht, dass mehrere Clientprozesse parallel dieselben Features verändern wollen und ebendies verhindern, wird oft als „pessimistisches" Locking bezeichnet, besonders im Gegensatz zum im Folgenden beschriebenen „optimistischen" Ansatz.

Es gibt die Möglichkeit, die Steuerung der konkurrierenden Zugriffe auf ein und dasselbe Feature mit einer „optimistischen" Strategie umzusetzen. Dieser Fall geht davon aus, dass es relativ selten passiert, dass zwei verschiedene Clientprozesse zeitgleich dieselben Daten editieren wollen – deshalb die Bezeichnung.

Es werden bei dieser Strategie keine harten Sperren gesetzt, vielmehr wird beim Locking ein Zeitstempel der zu sperrenden Features zurückgeliefert. Dieser Zeitstempel gibt an, wann zuletzt eines der betroffenen Feature verändert wurde. Die Lock-Operationen können in diesem Fall mehrmals durch unterschiedliche Clients und für dieselben Features aufgerufen werden. Die Umsetzung setzt voraus, dass der WFS oder die Datenhaltung intern für jedes Feature den Zeitpunkt der letzten Änderung pflegt. Die Konfliktbehandlung erfolgt erst innerhalb des Transaction Requests. Hier muss getestet werden, ob ein zu aktualisierendes Objekt nach Aufruf der Lock-Operation bereits durch einen zweiten Clientprozess verändert wurde (ob der Zeitstempel des Features nach dem Zeitstempel der in der LockId übergeben wird liegt). In diesem Fall schlägt die Transaktion fehl. Nur wenn zwischenzeitlich kein zweiter Prozess dieselben Daten verändert hat, kann die Transaktion ausgeführt werden.

Die Abbildung dieser „optimistischen" Strategie auf den WFS kann formal mit denselben Operationen arbeiten, wie im WFS-Standard vorgesehen. Die Operationen LockFeature bzw. GetFeatureWithLock bringen eine LockId zurück, die im „optimistischen" Fall ein Zeitstempel ist. Diese LockId wird dann bei der Transaction-Operation an den WFS übergeben.

Der Unterschied liegt darin, ob die Konfliktbehandlung bereits in der Lock-Operation erfolgt oder erst dann, wenn die Daten verändert werden. Ein Client, der einen „optimistischen" Server versorgt, kann nicht fest damit rechnen, dass ein Feature, das er gelockt hat, ihm zur Verfügung steht. Es kann passieren, dass die Veränderung abgewiesen wird, obwohl er ein Lock besitzt. Der mögliche Konflikt wird jedoch erkannt.

Clients müssen deshalb auf diese späte Konfliktbehandlung eingehen können. Dies ist allerdings keine wirklich zusätzliche Problematik, weil auch bei einer „pessimistischen" Strategie ein Veränderungsversuch durch Transaction fehlschlagen kann, z.B. wegen Zeitüberschreitung des Locks oder wegen eines Systemfehlers. Ein korrekter Client muss also letztlich auch in diesem Fall darauf vorbereitet sein, den gesamten Zyklus von Datenbeschaffung-Locking-Veränderung-Transaction zu wiederholen – auch wenn die Fälle beim „pessimistischen" Verfahren naturgemäß unwahrscheinlicher sind.

Wegen der geringen Unterschiede und der davon erwarteten Vorteile im Verhalten, bietet es sich an, für den OKSTRA-WFS auch die „optimistische" Strategie zuzulassen. Es ist aber festzuhalten, dass diese Art der Implementierung nur formal unter Verwendung der WFS-Operationen arbeitet. Sie entspricht nicht im vollen Umfang dem Wortsinn der WFS-Spezifikation.

7.4.3 Operationsübergreifende Transaktionen

Die Definition operationsübergreifender Transaktionen wäre für die effektive Umsetzung von fachlichen Diensten „über" dem WFS sehr wünschenswert. Sie würde die Programmierung stark vereinfachen, da das Verwendungsmuster aus jeweils einem Paar *Lock + Transaction* viel flexibler gestaltet werden könnte.

Technisch würde man das durch zwei zusätzliche Operationen machen, nämlich

1. *TransactionBegin*, welche einen *TransactionIdentifier* zurückbringt, und
2. *TransactionEnd* mit den Optionen *Commit* und *Abort* für einen *TransactionIdentifier*.

Der *TransactionIdentifier* wäre zusätzlich optionaler Input in die erweiterte *Transaction*-Operation.

Alternativ könnte man die beiden Operationen *TransactionBegin* und *TransactionEnd* auch als Optionen in die erweiterte *Transaction*-Operation aufnehmen.

Eins ist an dieser Stelle noch zu diskutieren:

Der WFS-Standard bietet auch die Operation *Native* an, die dazu verwendet werden kann, Basis-Funktionalität der zugrundeliegenden Datenbank anzusprechen. Das kann im Prinzip eingesetzt werden. Wir geben in diesem Falle aber die einheitliche Schnittstelle auf, da die zugrundeliegende Datenbank sicher durchaus verschieden ausfallen wird. Von diesem Weg ist deshalb eher abzuraten.

7.4.4 Verteilte Transaktionen

Bei serverübergreifenden Fortführungen weiten sich die beschriebenen Sachverhalte zu ernsthaften und nicht-trivialen Problemen aus, die durch Nutzung der WFS-Schnittstelle alleine nicht in den Griff zu kriegen sind.

Bei Durchführung einer komplexen Operation auf mehreren Servern kann nämlich die sichere Logik des Holens und Sperrens und anschließenden Veränderns der gesperrten Features durch eine *Transaction*-Operation nicht mehr funktionieren, weil für jeden Server eine extra Fortschreibungs-Operation durchgeführt werden muss.

Das Problem zeigt sich besonders auf zwei Arten:

1. Geht die erste *Transaction*-Operation auf dem ersten Server glatt und versagt die zweite, so bleibt das Gesamtsystem in einem inkonsistenten Zustand, der nicht zurückgefahren werden kann. Dies ist ein schlimmer Fall, weil die Datenbasis dauerhaft in einem insgesamt inkonsistenten und deshalb evtl. unbrauchbaren Zustand verbleibt.
2. Verbunden damit ist auch eine „unsaubere Sichtbarkeit" der nach der erfolgreichen *Transaction* auf dem ersten Server fortgeführten Features. Denn die dazu gehörenden Fortführungen auf Server 2 sind noch nicht ausgeführt, auch wenn dies schließlich erfolgreich geschehen sollte. Dies erzeugt bis zur Fertigstellung der Operation auf Server 2 temporär denselben inkonsistenten Zustand wie im ersten Fall.

Durch explizit kontrollierbare, also operationsüberspannende Transaktionen würden sich diese Effekte vermeiden lassen. Man würde in diesem Falle die Transaktion auf Server 1 offen lassen und „zurückrollen", falls die Transaktion auf Server 2 fehlschlägt. In diesem Falle würde der Client einen Teil der Rolle des Koordinators in einem verteilten Transaktionsprotokoll übernehmen.

Die für die Durchführung von verteilten Transaktionen erforderlichen Steuerungs- und Kommunikationsaufgaben sind leider i.d.R. kein standardisierter Bestandteil von Datenbankmanagementsystemen, sodass sich das Konzept des WFS einfach dieser Fähigkeiten bedienen könnte.

Soll eine solche Lösung tatsächlich umgesetzt werden, so muss die WFS-Schnittstelle um die Operationen des 2-Phasen-Commit-Protokolls (oder einer dessen Varianten) ergänzt werden. Dies dem Client zu überlassen, ist wahrscheinlich nicht anzuraten, da das System auch gegen den Ausfall des Clients geschützt werden muss.

7.5 Normalisierung des OKSTRA®

Für die Darstellung einiger Sachverhalte bietet der OKSTRA® die Möglichkeit alternativer Repräsentierungen. Dies ist für seine Rolle als Austauschformat zulässig, aber sehr hinderlich, wenn es darum geht, Abfragen gegen einen Datenbestand nach dem OKSTRA® XML Schema zu bilden: Es wäre dann erforderlich, die Abfrage für alle Alternativen zu formulieren, weil man nicht weiß, welcher Alternative der Datenbestand folgt. Ähnliches gilt für transaktionale (fortschreibende) Operationen, die sich naturgemäß an der vorhandenen Repräsentierung ausrichten müssen.

Es ist also nötig sicherzustellen, dass ein Sachverhalt nur auf eine Art dargestellt werden kann.

In folgenden Bereichen könnte/sollte eine Normalisierung vorgenommen werden:

1. **Schlüsseltabellen:** Die OKSTRA®-Schlüsseltabellen können nach dem OKSTRA® XML Schema wahlweise über XLink-Verweise referenziert oder jeweils an Ort und Stelle eingebettet werden. Dies kann man im Grunde offen lassen, da der Einsatz in Filterausdrücken in beiden Varianten dieselbe Syntax aufweist.
2. **Bereichsobjekte:** Die *Bereichsobjekte* des OKSTRA® (insbesondere die *Strassenbaudienststellen* und *Verwaltungsbezirke*) werden im Straßennetz auf *Netzbereichen* verortet. Diese *Netzbereiche* werden im OKSTRA® aus *Netzbereichskomponenten* zusammengesetzt, zu den die *Teilkanten* sowie deren Aggregationen (*Strecken* oder andere *Netzbereiche*) gehören. Es hat sich herausgestellt, dass sich in diesem Punkt keine normalisierte OKSTRA®-Darstellung finden lässt, die allen Implementierungen[4] gerecht wird. Es muss daher durch den OKSTRA-WFS das OKSTRA®-konform interpretierte, tatsächlich in den Straßendatenbanken vorhandene Modell angeboten werden. Das verwendete Modell ist in den Capabilities anzuzeigen.
3. **Streckenobjekte:** Die *Streckenobjekte* des OKSTRA® können sowohl auf einer *Teilkante* als auch auf einer Aggregation von *Teilkanten* (einer *Strecke*) verortet werden. Es soll nur eine dieser Strukturierungsarten verwendet werden.
4. **Teilkanten / Strecken / Netzbereiche:** Die Instanzen dieser Objektarten können im OKSTRA® zur Verortung einer beliebigen Menge von Fachobjekt-Instanzen dienen. Im Sinne einer Normalisierung wäre es jedoch sinnvoll, immer nur eine einzige Instanz eines Fachobjektes auf diesen Objektarten zu verorten, weil es sonst (bei im Grunde gleichem Dateninhalt) zu unterschiedlich strukturierten Netzverortungen kommen kann. Beispiel: Der Sachverhalt, dass zwei Fachobjekte im gleichen Stationsbereich auf demselben *Abschnitt* liegen, kann nach dem OKSTRA® entweder dadurch angegeben werden, dass die beiden

[4] Der grundsätzliche Unterschied besteht darin, dass bei einer der Implementierungen grundsätzlich alle Bereichsobjekte aus *Teilabschnitten* bestehen, während bei einer anderen bei den Bereichsobjekten *Strassenbaudienststelle* und *Verwaltungsbezirk* die *Netzbereiche* der untersten Ebenen aus *Teilabschnitten* und die Netzbereiche der höheren Ebenen aus den *Netzbereichen* der jeweils darunterliegenden Ebene zusammengesetzt werden.

Fachobjekte auf denselben *Teilabschnitt* verweisen oder dadurch, dass jedes Fachobjekt auf einen eigenen *Teilabschnitt* verweist, die aber beide identische Daten enthalten. Theoretisch könnte einer der beiden *Teilabschnitte* zusätzlich auch noch an einem *Netzbereich* beteiligt sein.

5. **Identifier**: Es ist ein einheitliches Konzept für die Bezeichnung von Features (Objektversionen) zu entwickeln. Es sollen serverübergreifende Referenzen möglich sein.
6. **GML-Geometrierepräsentation:** Das GML-Profil des OKSTRA® ermöglicht tlw. verschiedene Varianten zur Angabe von Geometrie. Hier ist ein möglichst einfaches Profil vorzusehen.
7. **Koordinatenreferenzsysteme:** Der OKSTRA® ermöglicht die Angabe von Koordinaten in verschiedenen Koordinatenreferenzsystemen. Das OKSTRA®-Dokument N0155 legt fest, welche Koordinatenreferenzsysteme in OKSTRA®-Objekten zulässig sind.
8. **Fachobjektarten:** Es ist nicht auszuschließen, dass im OKSTRA® – ähnlich wie bei den Netzverortungen (siehe Punkte 2 – 4 dieser Liste) auch bei einzelnen Fachobjektarten alternative Repräsentierungen existieren. Auch hier sollten erforderlichenfalls Normalisierungen vorgenommen werden.

Anforderung

In allen genannten Bereichen sind einschränkende Festlegungen zu treffen. Siehe hierzu die Abschnitte 8.3 und 9.3

7.6 OKSTRA®-Profile

OKSTRA®-Profile beschreiben eine Einschränkung des vollständigen OKSTRA®-Schemas, um z.B. bestimmte FeatureTypes auszublenden oder Kardinalitäten einzuschränken. Der genaue Umfang der Möglichkeiten zur Einschränkung des Schemas ist im Dokument T0009 „Festlegung von inhaltlichen Ausschnitten (Profilen) des OKSTRA®-Datenmodells" niedergelegt. Profile sollten verwendet werden, um die OKSTRA®-Version anzugeben sowie Einschränkungen zu 7.5, Punkte 2, 3, 7 und 8 zu vereinbaren.

Im Zusammenhang mit der Verwendung von OKSTRA®-Profilen ergibt sich jedoch in Bezug auf die *DescribeFeatureType*-Operation folgendes Problem:

Falls die Einschränkungen sich im von der Operation zurückgelieferten Schemadokument niederschlagen, ist dieses nicht mehr konform zum veröffentlichten OKSTRA®-Schema und dürfte daher streng genommen nicht im selben Namespace wie dieses liegen. Um jedoch die Validierbarkeit von XML-Dokumenten, die von einem OKSTRA-WFS abgegeben oder angenommen werden können, zu gewährleisten, soll für die Schemata, die von DescribeFeatureType erzeugt werden, dennoch der OKSTRA®-Namespace verwendet werden.

Anmerkung: Eine analoge Situation ergibt sich bei der Implementierung von Autorisierungsfunktionen. Auch hier ist erwünscht, dass DescribeFeatureType nur das Schema abliefert, das für den Benutzer unter seinen Berechtigungen tatsächlich sichtbar ist.

Anforderung

Ein OKSTRA-WFS soll Profile im Sinne des OKSTRA®-Dokuments T0009 „Festlegung von inhaltlichen Ausschnitten (Profilen) des OKSTRA®-Datenmodells" unterstützen. Es ist für jeden OKSTRA-WFS genau ein Profil anzugeben. Im Minimalfall enthält dieses nur die Angabe der unterstützten OKSTRA®-Version. Die im Capabilities-Dokument publizierte FeatureTypeList soll keine Feature Types beschreiben, die durch das Profil ausgeschlossen sind. DescribeFeatureType-Schema-Dokumente geben die durch das Profil (und ggf. durch die Autorisierung) bedingten Einschränkungen wieder.

7.7 Vordefinierte Standardabfragen

Die Spezifikation 4.2 sieht die Möglichkeit vor, dass ein WFS, und damit auch ein OKSTRA-WFS, vordefinierte Standardabfragen (Stored Queries) anbieten kann. Es ist überlegenswert, für bestimmte immer wiederkehrende und überall eingeführte Anwendungsfälle fachliche Standardabfragen zu formulieren. Solche Abfragen sind naturgemäß abhängig von dem Geschäftsprozess, in dem sie benötigt werden, und müssten deshalb in einem Abstimmungsprozess von einem Expertenteam erarbeitet werden. Dabei wäre auch festzulegen, ob eine bestimmte Standardabfrage verpflichtend oder optional sein soll.

Dieses Dokument spezifiziert die informationstechnischen Vorgaben für einen OKSTRA-WFS unter Berücksichtigung der fachlichen Vorgaben durch das OKSTRA®-Datenmodell. Die Spezifikation von fachlich determinierten Standardabfragen überschreitet daher den Rahmen dieses Dokumentes.

Die Mitbetrachtung fachlicher Aspekte würde bedeuten, dass es so etwas wie einen Straßennetz-WFS oder einen Entwurfs-WFS geben würde. Dafür müssten dann jeweils das zugehörige OKSTRA®-Profil sowie die zugehörigen Standardabfragen im Detail festgelegt werden.

Anforderung

Vor dem Aufsetzen eines jeden OKSTRA-WFS muss ein eigenes Dokument erstellt werden, in dem die Standardabfragen definiert und dokumentiert werden, die dieser OKSTRA-WFS enthalten soll.

7.8 Protokolle der Datenbereitstellungsschnittstelle

OGC-Dienste wie auch die WFS-Spezifikation benutzen im Allgemeinen http/POST und http/GET. Nur der POST-Schnittstelle ist es möglich, den gesamten Funktionsumfang abzubilden – sie muss daher angeboten werden.

GET ist oft sehr praktisch bei der Erstellung von Client-Software, ist aber nicht unbedingt erforderlich. Es ist aber zu erwarten, dass es zumindest für lesende Zugriffe zunehmend wichtiger werden wird.

Falls Authentifizierung und Autorisierung durch zusätzliche Komponenten unterstützt werden sollen, ist zu empfehlen, die in der Spezifikation vorgeschlagene SOAP-Schnittstelle verpflichtend zu machen, da sie es gestattet, in den Header-Feldern die erforderlichen Informationen zu übertragen, ohne die eigentlichen Verwendungsparameter zu stören.

Anforderung

Die WFS http/GET, http/POST- und SOAP-Protokolle sind verpflichtend.

7.9 Zusätzliche Anforderungen

Es hat sich erwiesen, dass die Implementierung von Geodateninfrastrukturen oft an der fehlenden Performanz der beteiligten Dienste scheitert.

Unter 500 Features pro Sekunde bei Queries und 100 Features pro Sekunde bei Insert/ Update/ Delete, sind solche Dienste nicht sinnvoll einsetzbar.

Gleichfalls hat sich herausgestellt, dass Dienste oft starke Einschränkungen bezüglich der abgebbaren Datenmengen und der Größe der Eingabedokumente bei schreibenden Operationen aufweisen, welche ihre Nutzung sehr erschweren oder unmöglich machen.

Authentifizierung und Autorisierung sind nicht Bestandteil des ISO/OGC-WFS-Konzepts. Authentifizierung setzt die Übertragung von Identitätsinformation von der nutzenden Anwendung zum OKSTRA-WFS voraus. Hierfür steht der internationale Standard Web Service Security (WSS) zur Verfügung. Auch für Autorisierungsinformation gibt mit XACML und GeoXACML Standards, allerdings gibt es vielfach bereits Autorisierungssysteme in Organisationsstrukturen (wie z.B. Behörden), die nicht übergangen werden können.

Anforderung
Der normative Teil des Profils wird um Bestimmungen zur Authentifizierung ergänzt. Zur Autorisierungsproblematik werden Empfehlungen ausgesprochen.

8 OKSTRA®-WFS lesend (normativ)

Die *Datenbereitstellungs-Schnittstelle für OKSTRA®-Daten auf der Basis des OGC Web Feature Service*, kurz *OKSTRA®-WFS*, wird in zwei aufeinander aufbauenden Spezifikationen als „Profil" des OGC WFS definiert.

In diesem Kapitel erfolgt die Definition des *OKSTRA®-WFS lesend,* welcher nur Funktionalität für den lesenden Zugriff auf die Daten bereitstellt.

Ein ***OKSTRA®-WFS lesend*** ist ein **OGC Basic WFS Version 2.0**, der die gesamte obligatorische Funktionalität des OGC Basic WFS aufweist, wie sie in der normativen Referenz 4.2, ergänzt durch Referenz 4.1, beschrieben ist, **und** der zusätzlich **alle** Anforderungen, aus dem vorliegenden Kapitel erfüllt.

Unter der obligatorischen Funktionalität des OGC Basic WFS wird hier die Funktionalität verstanden,

- welche im Standard durch entsprechende Verbformen wie „shall" oder „must", etc. ausdrücklich als solche bezeichnet wird **oder**
- für die keine Möglichkeit vorliegt, ihr Fehlen im Capabilities-Response-Dokument zu beschreiben.

Ausgenommen von dieser Regel sind alle Angaben, die im vorliegenden Dokument ausdrücklich anders geregelt sind.

Verbindliche Regelungen werden durch die Verbformen „muss" oder „soll" ausgedrückt. Konjunktivische Formen wie „sollte" dagegen drücken Empfehlungen aus.

Es werden die folgenden Namespaces benötigt und im Weiteren mit den angegebenen Abkürzungen bezeichnet:

Abkürzung	Namespace	Bedeutung
xlink	http://www.w3.org/1999/xlink	XLink-Schema
ows	http://www.opengis.net/ows/1.1	OWS Common Schema
fes	http://www.opengis.net/fes/2.0	Filter Encoding
wfs	http://www.opengis.net/wfs/2.0	Web Feature Service
gml	http://www.opengis.net/gml/3.2	GML
okwfs	http://www.okstra.de/okwfs/2.0	OKSTRA-WFS
okstra	http://www.okstra.de/okstra/xxxx	OKSTRA-Schema Version xxxx
okpro	http://www.okstra.de/profile/1.0	OKSTRA Profile

8.1 Profil des Web Feature Service

Dieser Teil der OKSTRA-WFS-Spezifikation für den *OKSTRA-WFS lesend* behandelt die zusätzlichen Definitionen, welche die OGC-WFS-Spezifikation betreffen.

Als erstes werden die bereitzustellenden Operationen des WFS im Einzelnen festgelegt. Darauf folgend werden Aussagen zu Koordinatensystemen, Identifiern, Ausnahmebehandlungen und Protokollen gemacht.

8.1.1 Operation: GetCapabilities

Die Operation GetCapabilities muss ein WFS_Capabilities-Dokument zurückbringen, das die Fähigkeiten des OKSTRA-WFS korrekt beschreibt.

Es sind alle Abschnitte außer *wfs:ServesGMLObjectTypeList* anzugeben. Im Einzelnen:

8.1.1.1 ows:ServiceIdentification

ows:ServiceType und ows:ServiceTypeVersion sind entsprechend als WFS und 2.0 auszuweisen.

Es wird empfohlen, bei ows:Title, ows:Abstract und ows:Keywords Angaben zu machen, welche den jeweiligen OKSTRA-WFS in Metadata Verzeichnissen gut unterscheidbar beschreiben.

8.1.1.2 ows:ServiceProvider

Keine Anforderungen oder Empfehlungen.

8.1.1.3 ows:OperationsMetadata

Es sind wenigstens die Operationen GetCapabilities, DescribeFeatureType, GetPropertyValue, GetFeature, ListStoredQueries und DescribeStoredQueries zu beschreiben.

Alle Operationen müssen unter ows:DCP/ows:http ein URL für ows:Get und ows:Post vorsehen.

Der Abschnitt 8.3.5.2 (Parameter domains) in 4.2 ist zu beachten. Diese Parameter können global für den ganzen Service oder lokal für einzelne Operationen angegeben werden.

Der Parameter mit Namen „srsName" ist anzugeben. Er soll alle in 8.1.8 geforderten und vom OKSTRA-WFS unterstützten Systeme mit ihren Codes gemäß 4.3.3 aufzählen. Es ist dabei die in 8.1.8 beschriebene urn-Notation zu verwenden.

Der Parameter mit Namen „outputFormat" ist anzugeben und soll zusätzlich das Format „application/xml+gml;version=3.2" angeben.

Bei Operationen, die den Standard-Parameter *resultType* unterstützen, soll dieser die Werte „results" und „hits" aufweisen.

Der Abschnitt 8.3.5.3 (Service and Operation Constraints) in 4.2 ist zu beachten. Die dort in Tabelle 13 aufgeführten Constraints ImplementsBasicWFS, KVPEncoding, XMLEncoding, SOAPEncoding, ImplementsStandardJoins, ImplementsSpatialJoins sollen den Wert TRUE haben.

Der globale Constraint mit Namen „CountDefault" soll nicht angegeben werden.

Der globale Constraint mit Namen „ResolveLocalScope" ist anzugeben und soll den Wert „*" aufweisen.

Falls der OKSTRA-WFS die Auflösung von „remote references" nicht unterstützt (das ist nicht gefordert), ist der globale Constraint mit Namen „ResolveRemoteScope" mit dem Werten „0" anzugeben. Falls „remote references" unterstützt werden, ist „ResolveRemoteScope" entsprechend den tatsächlich vorhandenen Fähigkeiten anzugeben.

Zusätzlich ist es erforderlich, als letztes Element von ows:OperationsMetadata das Element ows:ExtendedCapabilities anzugeben. Damit können folgende OKSTRA-WFS-spezifische Sachverhalte ausgedrückt werden:

- Das unterstützte OKSTRA®-Profil ist anzugeben. Die Angabe erfolgt entweder über eine URL per xlink:href-Attribute im OKSTRAWFSProfil-Element oder das Profil ist in der Profil-XML-Syntax als Inhalt des OKSTRAWFSProfil-Elements angegeben. Die eingebettete oder referierte Profilbeschreibung muss der Syntax des profil-Elements der Profilsprache entsprechen. Siehe hierzu 4.3.2. Das Profil muss wenigstens eine Versionseinschränkung vorgeben. Falls der OKSTRA-WFS Bereichs- und Streckenobjekte unterstützt, kann mit Hilfe des Profils auch die Struktur dieser Objekte vorgegeben und so die Anforderungen 7.5, Punkte 2 und 3 abgedeckt werden.

- Unterstützt die Implementierung *ImplicitGeometryFunctions*, wie sie in 8.2.1.3 beschrieben werden und gegebenenfalls welche?
- Ist im Falle eines *OKSTRA-WFS transaktional*, die *optimistische* Lockstrategie implementiert (Wert *optimistic*) oder die im OGC WFS-Standard vorgesehene *pessimische* (Wert *pessimistic*)? Die pessimistische Variante ist der Defaultwert.
- Werden im Falle eines *OKSTRA-WFS transaktional* nicht nur die geforderten lokalen Rückwärtsverweise unterstützt (Wert *backwards-only*) sondern auch Vorwärtsverweise (Wert *both-directions*)? „Nur Rückwärtsverweise" bilden den Defaultwert.

Die zu verwendende Syntax ist wie folgt:

```
<ows:OperationsMetadata>
…
    <ows:ExtendedCapabilities>
        <okwfs:OKSTRAWFSProfil>
            <profil name="wfsprofil" modellversion=2.016.0" />
        </okwfs:OKSTRAWFSProfil>
        <okwfs:ImplicitGeometryFunction>okwfs:punktAusPunktobjekt</okwfs:ImplicitGeometryFunction>
        <okwfs:ImplicitGeometryFunction> okwfs:linieAusStreckenobjekt</okwfs:ImplicitGeometryFunction>
        …
        <okwfs:LockingStrategy>optimistic</okwfs:LockingStrategy>
        <okwfs:LocalReferenceAllowedDirection>both-directions</okwfs:LocalReferenceAllowedDirection>

        …
    </ows:ExtendedCapabilities>
</ows:OperationsMetadata>
```

8.1.1.4 wfs:FeatureTypeList

Es sind alle vom OKSTRA-WFS bereitgestellten FeatureTypes zu beschreiben. Die anzugebenden FeatureTypes stellen eine echte oder unechte Teilmenge der OKSTRA®-FeatureTypes dar, die dem in den ExtendedCapabilities angegebenem OKSTRA®-Profil entsprechen.

Der Name des FeatureTypes ist bei wfs:Name anzugeben, z.B.:

```
<wfs:Name>okstra:Abschnitt</wfs:Name>
```

Es wird empfohlen, bei wfs:Title, wfs:Abstract und ows:Keywords Angaben zu machen, welche den jeweiligen FeatureType in Metadaten-Verzeichnissen gut unterscheidbar beschreiben.

Die Elemente wfs:DefaultSRS und wfs:OtherSRS sind anzugeben. Sie sollen alle in 8.1.8 geforderten und vom OKSTRA-WFS unterstützten Systeme aufzählen. Es ist dabei die in 8.1.8 beschriebene urn-Notation zu verwenden. wfs:NoSRS ist anzugeben, wenn der FeatureType keine Geometrie enthält.

wfs:OutputFormats muss „application/xml+gml;version=3.2" enthalten.

Das Element wfs:WGS84BoundingBox ist anzugeben, wenn es sich um einen geometrietragenden FeatureType handelt oder um einen solchen mit impliziter Geometrie. Es soll das einschließende Rechteck der Geometrien in Grad angegeben werden. Reihenfolge der Koordinaten: Geographische Länge – Geographische Breite.

8.1.1.5 fes:Filter_Capabilities

Folgende GeometryOperands sind mindestens anzugeben:

gml:Envelope, gml:Point, gml:LineString, gml:Polygon

Folgende Conformance-Prädikate sind als true anzugeben:

ImplementsQuery, ImplementsAdHocQuery, ImplementsMinStandardFilter, ImplementsStandardFilter, ImplementsMinSpatialFilter

Folgende SpatialOperators sind mindestens anzugeben:

BBOX, Disjoint, Intersects, Within, Contains, Overlaps, Dwithin

LogicalOperators und alle im Filter Encoding (Referenz 4.1) spezifizierten ComparisonOperators sind anzugeben.

In den AvailableFunctions sind die arithmetischen Funktionen Add, Sub, Mul, Div zu definieren.

Von den Id_Capabilities ist wenigstens fes:EID anzugeben.

8.1.2 Operation: DescribeFeatureType

Die Operation DescribeFeatureType erzeugt ein Schemaprofil des OKSTRA®-XML-Schemas für einen oder mehrere FeatureTypes. Die Operation erzeugt ein XML-Schema-Dokument, oder im Fehlerfall ein Exception-Dokument.

Es ist zu beachten, dass die Bildung solcher Schemaprofile im selben Namespace (okstra-Namespace), wie durch DescribeFeatureType gefordert, im Grunde nicht durch den XML Schema Standard abgedeckt ist, da ein Client beim Eintreffen eines Dokuments zu einem Namespace grundsätzlich annehmen darf, dass dieses das korrekte und vollständige Schema zum Namespace beinhaltet. Er darf den Inhalt „cachen" und in Zukunft als das gesamte Schema zum Namespace ansehen. Wenn OKSTRA-WFS-Clients so vorgehen, kann dies zu Fehlersituationen führen.

Im Folgenden wird im Einzelnen das Vorhandensein und die Wirkung einzelner Bestandteile der DescribeFeatureType-Operation beschrieben:

8.1.2.1 wfs:DescribeFeatureType/@outputFormat

Ein OKSTRA-WFS soll „application/xml+gml;version=3.2" unterstützen.

Das Response-Dokument soll ein XML-Schema-Dokument sein, das nur die Definitionen der durch TypeName geforderten FeatureTypes enthält und ansonsten möglichst nur die dafür benötigten zusätzlichen Schemafragmente.

8.1.2.2 wfs:DescribeFeatureType/wfs:TypeName

Es sollen alle in den Capabilities angebotenen FeatureTypes unterstützt werden.

8.1.3 Standard-Parameter

Es gibt in WFS 2.0 eine Reihe von Standard-Parametern, die von verschiedenen Operationen unterstützt werden. Die detaillierte Spezifikation ist in 4.2, Abschnitt 7.6 ausgeführt. Hier folgt eine kurze Erläuterung der Semantik der Parameter (in XML sind das Attribute)

8.1.3.1 @startIndex

Gibt den Startindex an, ab dem Features aus der intern hergestellten Resultatmenge einer Abfrage ausgegeben werden sollen. Kann im Zusammenwirken mit count dazu verwendet werden, die Resultatmenge stückweise auszulesen. Unterstützung im OKSTRA-WFS ist verpflichtend.

8.1.3.2 @count

Gibt die Anzahl Features an, die aus der intern hergestellten Resultatmenge einer Abfrage ausgegeben werden sollen. Kann im Zusammenwirken mit startIndex dazu verwendet werden, die Resultatmenge stückweise auszulesen. Unterstützung im OKSTRA-WFS ist verpflichtend.

8.1.3.3 @resultType

Werte sind result und hits. result liefert eine Menge von Features nach einer Abfrage, hits nur deren Anzahl. Unterstützung im OKSTRA-WFS ist verpflichtend.

Bei „hits" sollen nur die primär durch Query-Konstrukte gefundenen Features gezählt werden. Zusätzliche Features, welche möglicherweise durch Verfolgung von Referenzen (siehe *resolve*) sollen nicht gezählt werden.

8.1.3.4 @outputFormat

Default und verpflichtend ist application/gml+xml;version=3.2. Unterstützung weiterer Formate ist optional, dann muss dieser Parameter ausgewertet werden.

8.1.3.5 @resolve

Gibt an, wie die Auflösung von Referenzen eines abgefragten Features zu anderen Features gehandhabt werden soll. Werte sind:

- local: nur lokale Referenzen in derselben Datenhaltung werden aufgelöst.
- Remote: nur entfernte Referenzen in anderen Datenhaltungen werden aufgelöst.
- all: alle Referenzen werden aufgelöst.
- none: keine Referenzen werden aufgelöst.

local muss wenigstens von einem OKSTRA-WFS unterstützt werden. Falls andere Optionen aus der Liste erlaubt sind, muss der Parameter ausgewertet werden.

8.1.3.6 @resolveDepth

Default ist „beliebig tief". Es werden rekursiv alle Properties aller durch Query-Konstrukte sich qualifizierender Features bis zur angegebenen Tiefe verfolgt. Bei der Verfolgung wird der Wert des Parameters resolve beachtet. Unterstützung im OKSTRA-WFS ist verpflichtend.

8.1.3.7 @resolveExpiry

Der Wert des Attributs kann ignoriert werden, soweit keine datenhaltungsübergreifende Verfolgung von xlink:href implementiert ist.

8.1.3.8 @inputFormat

Default und verpflichtend ist application/gml+xml;version=3.2. Unterstützung weiterer Formate ist optional, dann muss dieser Parameter ausgewertet werden.

Der Parameter wird in schreibenden Operationen verwendet.

8.1.3.9 @srsName

Der Parameter gibt das Default-Koordinatenreferenzsystem an, das in schreibenden Operationen anzunehmen ist, wenn keine Angabe dazu an Geometrie hängt.

8.1.4 Query-Parameter

Jedes Query-Element führt entsprechend seiner Parametrisierung durch die im Folgenden beschriebenen Elemente und Attribute zu einer Suchoperation auf dem Datenbestand.

Query-Elemente werden verwendet in den Operationen GetPropertyValue, GetFeature, GetFeatureWithLock und LockFeature.

Die gefundenen Features werden in der Reihenfolge der Query-Elemente in der wfs:FeatureCollection abgelegt. Treten infolge sich überschneidender Queries Feature-Exemplare mehrfach auf, so wird nur das erste Feature explizit im gml:featureMember eingetragen. Bei allen weiteren wird das xlink:href-Konstrukt eingesetzt, um auf das erste Auftreten zu verweisen.

Im Folgenden werden die im Query-Element möglichen Attribute und Elemente kurz beschrieben.

8.1.4.1 @typeNames

Das Attribut gibt die FeatureTypes an, die gesucht werden sollen, z.B. (okstra:Abschnitt)(okstra:Ast). Die Implementierung ist verpflichtend.

8.1.4.2 @aliases

Dieses Attribut gestattet die Vergabe von Aliasnamen für typeNames (analog zu Aliasnamen in SQL Select). Die im OGC-WFS-Dokument angedeutete Syntax für *Joins* und *Alias*-Definitionen ist im Standard nicht hinreichend für eine Implementierung spezifiziert. Es ist nicht erforderlich, diese Funktionalität zu implementieren.

8.1.4.3 @srsName

Die Angabe eines der unterstützten (siehe 8.1.8) Koordinatensysteme bewirkt die Ausgabe der Geometrien in diesem angegebenen Koordinatensystem. Die Implementierung ist verpflichtend.

8.1.4.4 @handle

Die Angabe des handle-Attributs dient zur verbesserten Zuordnung von Exceptions zu den einzelnen Query-Angaben. Die Implementierung ist verpflichtend.

8.1.4.5 wfs:Query/PropertyName

Die im OGC WFS vorgesehene Möglichkeit zur Einschränkung der Struktur der einzelnen Features auf explizit angegebene PropertyNames ist zu unterstützen. Qua Schema obligatorische Properties sind immer auszugeben.

8.1.4.6 wfs:Query/fes:Filter

Das Filterelement soll entsprechend den Angaben in 8.1.10 interpretiert werden.

8.1.4.7 wfs:Query/SortBy

Ein OKSTRA-WFS braucht SortBy nicht zu unterstützen.

8.1.5 Operation: GetPropertyValue

Die Operation GetPropertyValue erfragt den Wert einer einzelnen (einfach oder multipel vorhandenen) Eigenschaft aus einer Menge von Features. Diese Menge wird über ein Query-Element ausgewählt. Der verpflichtende ValueReference-Parameter gibt an, wie von den Features der abgefragten Menge zu der interessierenden Property zu navigieren ist.

Die Menge der so gefundenen Werte wird im Response-Dokument nachgewiesen, im Fehlerfall gibt es stattdessen ein Exception-Dokument.

8.1.6 Operation: GetFeature

Die Operation GetFeature erfragt OKSTRA®-Objekte (Features), welche sich entsprechend den Angaben im GetFeature-Request qualifizieren. Die Operation bringt ein gml:FeatureCollection-Dokument zurück, oder im Fehlerfall ein Exception-Dokument.

Das wfs:FeatureCollection-Document enthält wenigstens

1. das gml:boundedBy-Property und
2. null oder mehr gml:featureMember-Properties mit OKSTRA®-Objektversionen entsprechend der Definition von OKSTRA®-XML.

8.1.7 Vordefinierte Abfragen

Es sind verpflichtend die Operationen ListStoredQueries und DescribeStoredQueries.

ListStoredQueries liefert eine Liste von Kurzbeschreibungen aller vorhandenen vordefinierten Abfragen, die jeweils eine textliche Beschreibung, den ausgegebenen FeatureType sowie eine eindeutige Identifikation enthalten.

Mit Hilfe der Identifikation einer vordefinierten Abfrage kann man über DescribeStoredQueries die detaillierten Metadaten zu den Abfragen abholen.

Jeder WFS 2.0 muss eine vordefinierte Abfrage anbieten, die ein einzelnes Feature über seine eindeutige ID abliefert.

8.1.8 Koordinatenreferenzsysteme

Von einem OKSTRA-WFS können alle im Dokument N0155 (4.3.3) aufgeführten, für den OKSTRA® vorgesehenen Koordinatensysteme verwendet werden.

Es müssen mindestens unterstützt werden:

Name	Datum	Ellipsoid	Projektion	Koordinaten-reihenfolge	Code gemäß 4.3.3
ETRS89	ETRS89	GRS80	Keine (geographisch)	North-East	ETRS89_Lat-Lon
WGS84	WGS84	WGS84	Keine (geographisch)	North-East	WGS84_Lat-Lon
DHDN-GKz z=2,3,4,5*)	Potsdam	Bessel	Transverse Mercator	North-East	DE_DHDN_3GKz
ETRS-UTM32N	ETRS89	GRS80	Transverse Mercator	East-North	ETRS89_UTM32

*) Nur die Gauß-Krügerstreifen, in denen die Daten liegen, sind verpflichtend.

Die Koordinatensysteme können in allen Operationen eines OKSTRA-WFS verwendet werden, z.B. in Filter-Angaben oder als gewünschtes System beim Erfragen von Features.

Zur Spezifikation von Koordinatensystemen soll an allen Stellen die urn-Schreibweise

urn:adv:def:crs:*AdV-Kurzbezeichnung*

eingesetzt werden, wie im Dokument N0155 beschrieben.

Beispiel:

```
<gml:Point srsName= "urn:adv:def:crs:ETRS89_UTM32">
…
</gml:Point>
```

8.1.9 Identifier, gml:id und xlink:href Verweise

Objekt-Identifier für einen OKSTRA-WFS werden verwendet zur Bezeichnung von Features (OKSTRA®-Objektversionen). Sie treten auf im gml:id-Attribut und in xlink:href-Referenzen.

Objekt-Identifier müssen eindeutig und stabil sein.

Um einen aufwändigen Registrierungsmechanismus zur Separierung der Namensräume der OKSTRA-WFS zu vermeiden, wird vorgeschlagen, die Identifier als http-URLs auszulegen, da eine Institution, die einen OKSTRA-WFS betreibt, sowieso die alleinige Kontrolle über ihre Internetdomäne besitzt und innerhalb dieser die Namen beliebig vergeben kann.

Insgesamt bestehen Objekt-Identifikatoren eines OKSTRA-WFS dann aus drei Teilen wie folgt:

1. *lokal-eindeutiger-identifier* steht für einen stabilen und innerhalb des FeatureTypes eindeutigen Identifier für Features (also bei Historisierung für Objektversionen), welchen die jeweilige Datenhaltung vergibt.
2. *feature-type-name* ist der Klartext-Name des FeatureTypes, der bezeichnet wird.
3. *namensraum-präfix* bezeichnet den logischen OKSTRA-WFS-Service, welcher das Feature beherbergt. Es soll sich nicht um den tatsächlich installierten Server handeln, weil sonst beim Umzug des Service auf andere physische oder virtuelle Systeme die Referenzen ungültig werden. Die Abbildungen der logischen auf die installierten Systeme können in der Webserver-Software durch rewrite-Regeln implementiert werden.

Die drei Teile werden durch / getrennt.

Das gml:id-Attribut werden damit wie folgt geschrieben:

gml:id="*namensraum-präfix*/ *feature-type-name*/ *lokal-eindeutiger-identifier*"

Dieselbe Schreibweise wird auch bei Objekt-Referenzen xlink:href eingesetzt. Beispiel:

xlink:href="http://www.bast.de/bisstrawfs/Netzknoten/5608003"

So können serverübergreifende Referenzen dokumentiert werden.

8.1.10 Ausnahmebehandlung

Beim Auftreten von Fehlersituationen erzeugt der OKSTRA-WFS ein ExceptionReport-Element entsprechend dem OGC-WFS-Standard anstelle der normalen Rückgabe. Bei Verwendung des SOAP-Protokolls sind die hierzu gemachten Aussagen im OGC-WFS-Standard zu beachten.

Codes und Fehlertexte sind, obwohl dies wünschenswert wäre, z.Z. im WFS-Standard nicht festgelegt. Es folgt deshalb hier für den OKSTRA-WFS eine Festlegung der möglichen Werte für das Attribut

exceptionCode="..."

am Element <Exception>.

Der zugehörige <ExceptionText> ist wird hier nicht wörtlich festgelegt. Er soll jedoch in deutscher Sprache angegeben werden und so aussagekräftig formuliert werden, dass die Fehlersituation damit klar beschrieben ist.

exceptionCode="..."	**Bedeutung**
OperationNotSupported	Die verlangte Operation entspricht dem WFS-Standard, wird aber nicht unterstützt.
MissingParameterValue	Ein Parameter für die verlangte Operation hat keine Wertangabe, und es existiert kein Default dafür.
InvalidParameterValue	Der Wert für einen Parameter der verlangten Operation ist ungültig.
VersionNegotiationFailed	Die Versionsliste im AcceptVersions-Parameter im GetCapabilities-Request enthält keine Version, die von diesem Server unterstützt wird.
InvalidUpdateSequence	Der Wert des optionalen updateSequence-Parameters im GetCapabilities-Request ist größer als der aktuelle Wert der UpdateSequence-Nummer in den Metadaten

OptionNotSupported	Eine im Request angegebene Option wird von diesem Server nicht unterstützt.
CannotLockAllFeatures	Eine LockOperation mit LockAction=ALL konnte nicht alle Features sperren.
DuplicateStoredQueryIdValue	Die für die gespeicherte Abfrage angegebene Id existiert bereits.
DuplicateStoredQueryParameterName	Der Name eines Parameters für die gespeicherte Abfrage wird bereits verwendet.
FeaturesNotLocked	Eine Transaction-Operation möchte Features verändern oder löschen, die nicht vorher gesperrt wurden.
InvalidLockId	Die in der Transaction-Operation angegebene LockId wurde nicht von diesem Server erzeugt.
InvalidValue	Die Transaction-Operation versucht den Wert einer Eigenschaft zu verändern, die nicht dem GML-Schema entspricht.
LockHasExpired	Die Gültigkeit der in der Transaction-Operation angegebenen LockId ist abgelaufen.
OperationParsingFailed	Die verlangte Operation ist syntaktisch falsch.
OperationProcessingFailed	Die Ausführung der verlangten Operation ist fehlgeschlagen.
ResponseCacheExpired	Die im Server vorgehaltene Ergebnisliste zum stückweisen Auslesen ist wegen Zeitablauf nicht mehr verfügbar.
FeatureTypeNotSupported	Der verlangte FeatureType wird nicht unterstützt.
SRSNotSupported	Das verlangte Koordinatenreferenzsystem wird nicht unterstützt.
ParameterInconsistency	Es wurden inkonsistente oder widersprüchliche Parameter angegeben, z.B. bei GET FILTER= zusammen mit BBOX=
InvalidPropertyName	Der angegebene PropertyName entspricht nicht dem OKSTRA®-Schema oder ist syntaktisch ungültig.
InvalidLiteralValue	Der Wert einer Konstante in einem Filterausdruck ist unzulässig.
InvalidGeometryLiteralValue	Der Wert einer Geometriekonstanten ist unzulässig.
FilterOperatorNotAvailable	Der verlangte Operator in einem Filterausdruck wird nicht unterstützt.
NoApplicableCode	Ein Fehler, der sich nicht sinnvoll auf die oben angegebenen Codes abbilden lässt.

Es ist zu erwarten, dass sich aus der Implementierung und in der Praxis zusätzliche Fehlersituationen ergeben werden, welche durch die oben angegebene Liste nicht adäquat abgedeckt sind. Die-

se sind zunächst auf *NoApplicableCode* abzubilden. In einer zukünftigen Überarbeitung der OKSTRA-WFS-Spezifikation sind diese Fehlersituationen dann in die obige Liste einzuarbeiten.

8.1.11 Protokolle

Der *OKSTRA®-WFS lesend* muss für alle unterstützten Operationen die Protokolle http/POST, http/GET und SOAP/POST in vollem Umfang unterstützen.

8.2 Profil des Filter Encoding

Das Filter Encoding für den OKSTRA-WFS wird nahezu vollständig als Implementierung vorausgesetzt. Es werden aufgrund der starken relationalen Vernetzung des OKSTRA® sogar weitreichende Erweiterungen, besonders beim Zugriff auf Properties, gefordert.

Im Einzelnen:

8.2.1 Property-Zugriffe

8.2.1.1 Erweiterte XPath-Semantik

In Erweiterung der Semantik der geforderten XPath-Notation im Element fes:ValueReference in Referenz 4.1, soll diese beim OKSTRA-WFS auch Anwendung finden, um auf Eigenschaften zugreifen zu können, die referenziell (über xlink:href) angebunden sind.

Es ist dabei syntaktisch unerheblich, ob ein objekthafter Property-Wert eingebettet vorliegt oder über xlink:href angebunden ist.

Ein solcher Zugriff repräsentiert in seiner Gesamtheit die Menge der so vom Objekt (Feature) aus erreichten Werte.

8.2.1.2 Allgemeine Prädikate

Die Unterstützung allgemeiner Prädikate ist in der Spezifikation zum Filter Encoding in der Version 2.0 zwingend vorgeschrieben.

8.2.1.3 Geometrische Funktionen

Falls die vom OKSTRA-WFS unterstützte OKSTRA®-Version keine implizite Geometrie für Netzbezüge unterstützt, wird empfohlen zusätzlich geometriewertige Funktionen, genannt *ImplicitGeometry-Functions*, anzubieten, welche die aus dem Netzbezug implizit vorhandene Geometrie bereitstellen.

Alle Funktionen berechnen aus dem gegebenen „nodeset", welches je nach Funktion bestimmte Features repräsentieren muss, und einem Stichtag eine gml:MultiGeometry bestimmter Dimensionalität.

Signatur	**Resultattyp**	**Einschränkung für nodeset**
okwfs:punktAusPunktobjekt(nodeset,stichtag)	MultiPoint	Punktobjekt
okwfs:linieAusStreckenobjekt(nodeset,stichtag)	MultiCurve	Streckenobjekt
okwfs:strassenachse(nodeset,stichtag)	MultiCurve	Strasse

Das nodeset kann fehlen. In diesem Falle wird der Kontext herangezogen, welcher dann den Einschränkungen genügen muss.

Das Argument „stichtag" steht für ein Datum, gegeben als Text in ISO8601-Format. Es wählt von den möglichen Geometrien in einer Relation zwischen historisierten Objekten die zum Stichtag gültigen aus. Das Argument kann fehlen, wenn ein Stichtag auf andere Weise bekannt ist.

Falls diese Funktionen zur Verfügung stehen, müssen sie der hier vorgegebenen Benennung folgen und in den ows:ExtendedCapabilities (siehe 8.1.1.3) erklärt sein.

8.2.1.4 schema-element()

Diese Funktion muss nicht unterstützt werden, da das OKSTRA®-XML-Schema flach ist, d.h. ohne Vererbung auskommt.

8.2.2 Geometrische Prädikate

Folgende in 4.1 definierte Prädikate sind obligatorisch für einen OKSTRA-WFS:

- BBOX
- Disjoint
- Intersects
- Within
- Contains
- Overlaps
- DWithin

Für den Fall, dass der Property-Zugriff auf die Geometrie über XPath-Notation eine Menge von Geometrien beschreibt, so werden diese im Prädikat implizit zu einer Collection zusammengefasst.

8.2.3 Skalare Vergleiche

Alle in 4.1 definierten skalaren Vergleiche sind obligatorisch für einen OKSTRA-WFS.

Für den Fall, dass der Property-Zugriff auf den Wert eines Operanden eine Menge von Werten beschreibt, so wird der Vergleich gemäß der Angabe des *matchAction*-Attributes des Operators durchgeführt. Ist das Attribut nicht vorhanden gilt der Default any (Existenzquantifizierung - „es gibt ein...").

Der Datentyp des Vergleichs ergibt sich aus dem Typ des teilnehmenden Property-Zugriffs gemäß dem OKSTRA®-Schema. Vergleiche werden durchgeführt für numerische Größen, Texte und DateTime-Angaben.

8.2.4 Logik

fes:And, fes:Or und fes:Not sind zu unterstützen.

8.2.5 Ausdrücke, Literale, Funktionen

Literale können Texte, numerische Größen, Date/DateTime-Größen (in ISO8601-Syntax) und Geometrien (in GML-Syntax) enthalten.

fes:Function wird im OKSTRA-WFS wenigstens eingesetzt, um die Grundrechenarten Add, Sub, Mul und Div anzubieten. Diese und evtl. weitere sind in den Filter_Capabilities zu erklären.

8.2.6 Identifier-Filter

Bezüglich der Filterung nach Objektidentifiern ist die Unterstützung des Id-Operators (4.1, Abschnitt 7.11) verpflichtend. Die für die Versionsunterstützung hierbei zulässigen Attribute sollen ignoriert werden.

8.3 Einschränkungen im OKSTRA®-Schema

Dieser Abschnitt beschreibt Normalisierungen im OKSTRA®-Schema, welche

- bei der Formulierung von Filterausdrücken – insbesondere beim Zugriff auf Eigenschaften über XPath – vorausgesetzt werden dürfen und
- welche bei der Abgabe von OKSTRA®-Daten durch den OKSTRA-WFS eingesetzt werden.

8.3.1 Schlüsseltabellen

Ein OKSTRA-WFS hat die Freiheit, die Inhalte von Schlüsseltabellen entweder in die Properties einzubetten oder als getrennte Objekte im GetFeature-Response aufzuführen und diese aus den Properties per xlink:href zu referenzieren.

8.3.2 Strecken- und Bereichsobjekte

Da kein normalisiertes OKSTRA®-Profil gefunden werden kann, das allen Implementierungen gerecht wird, soll durch den OKSTRA-WFS das OKSTRA®-konform interpretierte, tatsächlich in den Straßendatenbanken vorhandene Modell angeboten werden. Mit Hilfe der OKSTRA®-Profile können für Relationen, die eine Objektart mit Unterklassen zum Ziel haben, die tatsächlich als Partner zugelassenen Unterklassen eingeschränkt werden. Diesen Mechanismus kann man auf die Objektarten Netzbereichskomponente, Streckenkomponente und Teilkante anwenden. Um für verschiedene Bereichsarten verschiedene Schachtelungsschemata auszudrücken, muss man die Möglichkeit der Profile ausnutzen, Eigenschaftsschachtelungen zu berücksichtigen. Die für die Schachtelungshierarchie der Strecken- und Bereichsobjekte letztlich atomaren Teilkanten sollen nur für genau eine lineare Eigenschaft verwendet werden.

8.4 Allgemeine zusätzliche Forderungen

8.4.1 Performanz

Es wird empfohlen, dass Queries an einen OKSTRA-WFS mit mindestens 500 Features pro Sekunde im Durchschnitt beantwortet werden.

8.4.2 Größenbeschränkungen

Es wird empfohlen, die Implementierung eines OKSTRA-WFS so auszulegen, dass keine Beschränkungen bezüglich der Größe von Request- oder Response-Dokumenten bestehen, z.B. der Größe der erfragbaren FeatureCollections.

9 OKSTRA®-WFS transaktional (normativ)

In diesem Kapitel erfolgt die Definition des *OKSTRA®-WFS transaktional,* welcher die Funktionalität des *OKSTRA®-WFS lesend* auf schreibende Zugriffe erweitert.

Ein ***OKSTRA®-WFS transaktional*** ist entweder

> ein **OGC Transactional WFS Version 2.0**, der die gesamte obligatorische Funktionalität des *OKSTRA®-WFS lesend* aufweist, wie sie in Kapitel 8 dargelegt wurde, **und** der zusätzlich alle Anforderungen aus dem vorliegenden Kapitel erfüllt, außer denen aus 9.1.6, 9.1.7 und 9.1.8.2. Er muss außerdem AutomaticDataLocking unterstützen (Referenz 4.2, Tabelle 14).

oder

> ein **OGC Locking WFS Version 2.0**, der die gesamte obligatorische Funktionalität des *OKSTRA®-WFS lesend* aufweist, wie sie in Kapitel 8 dargelegt wurde, **und** der zusätzlich **alle** Anforderungen aus dem vorliegenden Kapitel erfüllt.

Die verwendeten Notationen entsprechen denen im Kapitel 8.

9.1 Profil des Web Feature Service

Dieser Teil der OKSTRA-WFS-Spezifikation für den *OKSTRA-WFS transaktional* behandelt die zusätzlichen, über die im Abschnitt 8.1 hinaus vorzunehmenden Definitionen, welche die OGC-WFS-Spezifikation betreffen.

Als erstes werden die Operationen des WFS behandelt (GetCapabilities, DescribeFeatureType, GetFeature, GetFeatureWithLock, LockFeature, Transaction) und im Einzelnen festgelegt.

Die Aussagen zu Koordinatensystemen, Identifier, Ausnahmebehandlung und Protokollen sind wie bei *OKSTRA-WFS lesend.*

9.1.1 Operation: GetCapabilities

Im Folgenden werden die Unterschiede bezüglich des WFS_Capabilities-Dokuments dargelegt.

9.1.1.1 ows:ServiceIdentification

Identisch.

9.1.1.2 ows:ServiceProvider

Identisch.

9.1.1.3 ows:OperationsMetadata

Es sind wenigstens die Operationen GetCapabilities, DescribeFeatureType, GetPropertyValue, GetFeature, ListStoredQueries, DescribeStoredQueries und Transaction zu beschreiben. Ist explizite Isolation der Schreibzugriffe implementiert, sind zudem GetFeatureWithLock und LockFeature zu beschreiben. Ist implizite Isolation bei Transaction vorgesehen, ist der Operation Constraint AutomaticDataLocking anzugeben.

Alle Operationen müssen unter ows:DCP/ows:http ein URL für ows:Get und ows:Post vorsehen.

Bei GetFeatureWithLock sind die Standard-Parameter wie bei GetFeature anzugeben.

LockFeature soll als *lockAction* „ALL" und „SOME" angeben.

Bei der Operation Transaction ist wenigstens *inputFormat* „application/xml+gml;version=3.2.1" anzugeben. Bei *releaseAction* soll „ALL" und „SOME" angeben sein.

Zusätzlich ist es erforderlich, als letztes Element von ows:OperationsMetadata das Element ows:ExtendedCapabilities anzugeben. Siehe hierzu 8.1.1.3.

9.1.1.4 wfs:FeatureTypeList

Der Abschnitt ist wie bei *OKSTRA-WFS lesend* zu erzeugen.

Zusätzlich gilt:

Wenn das wfs:Operations Element angegeben wird, so soll es für die FeatureTypes, welche zum Schreiben freigegeben sind, wenigstens

```
<wfs:Operation>Query</wfs: Operation>
<wfs:Operation>Insert</ wfs: Operation>
<wfs:Operation>Update</ wfs: Operation>
<wfs:Operation>Replace</ wfs: Operation>
<wfs:Operation>Delete</ wfs: Operation>
<wfs:Operation>Lock</ wfs: Operation>
```

enthalten.

9.1.1.5 fes:Filter_Capabilities

Identisch.

9.1.2 Operation: DescribeFeatureType

Identisch.

9.1.3 Operation: GetPropertyValue

Identisch.

9.1.4 Operation: GetFeature

Identisch.

9.1.5 Vordefinierte Abfragen

Identisch.

9.1.6 Operation: GetFeatureWithLock

Die Operation GetFeatureWithLock ist weitgehend identisch zu GetFeature.

Es wird versucht, ein Lock auf alle selektierten Feature-Instanzen zu nehmen, einschließlich derer, welche durch XLink-Verfolgung automatisch hinzugetreten sind. Falls dies gelingt, wird das vergebene Lock im Attribut *lockId* der wfs:FeatureCollection nachgewiesen. Falls das Lock nicht genommen werden kann, wird eine Ausnahme erzeugt.

Zum Nachweis von Locks siehe 9.1.6.2.

9.1.6.1 wfs:GetFeatureWithLock/@outputFormat

Aufgrund des zwingenden Nachweises des Locks über das lockId-Attribut, ist nur das outputFormat „application/xml+gml;version=3.2.1" für GetFeatureWithLock geeignet. Andere Formate sollten deshalb nicht angeboten werden.

9.1.6.2 wfs:GetFeatureWithLock/@expiry

Das Attribut gibt an, nach wie vielen Minuten das Lock (vertreten durch das LockId) automatisch aufgegeben wird, falls es nicht vorher benutzt wurde. Der Defaultwert ist „5".

9.1.7 Operation: LockFeature

Die Operation LockFeature versucht Locks für spätere schreibende Zugriffe auf selektierte Feature-Instanzen zu nehmen. Die Selektion erfolgt über einen Filter. Die Operation kann so parametrisiert werden, dass entweder

- alle dem Filter genügenden Features gelockt werden, wobei die Operation scheitert, wenn einige der qualifizierenden Features bereits andere Locks besitzen, oder
- nur die Features ein Lock erhalten, welche noch kein anderweitiges Lock aufweisen. Diese Variante scheitert nur, wenn alle qualifizierenden Features bereits vorher gelockt sind.

Bei Erfolg wird das Lock durch eine LockId im Response der Operation nachgewiesen. Bei Nichterfolg wird eine Ausnahme erzeugt.

Die Locks werden nur für eine gewisse Zeit gewährt, die im Request-Paket durch den Parameter *expiry* angegeben werden kann. Danach verfallen die nachgewiesenen LockIds – ihre Verwendung in schreibenden Operationen führt dann zu einer Ausnahmebehandlung.

Die Locks in der Spezifikation des OGC-WFS sind von ihrer Funktionalität her als klassische, exklusive („pessimistische") Locks aus der Datenbanktechnologie gemeint. Während der Lebensdauer der Locks steht das Feature für schreibende Operationen ausschließlich dem Client zur Verfügung, der Kenntnis der „LockId" hat.

Für den OKSTRA-WFS sollen neben diesen „pessimistischen" Locks auch „optimistische" Locks zugelassen werden. In diesem Falle wird durch LockFeature der jüngste (also von der Zeitachse her größte) Zeitstempel aller sich qualifizierenden Features ermittelt und im LockId nachgewiesen. Bei nachfolgenden schreibenden Operationen wird geprüft, ob der Zeitstempel des Features kleiner oder gleich dem LockId ist. Ist er es nicht, wird die schreibende Operation (die gesamte Transaction-Operation) zurückgewiesen.

Die „optimistische" Locking-Strategie muss zwingend im Capabilities-Response-Dokument (in ows:OperationsMetadata/ows:ExtendedCapabilities) des OKSTRA-WFS bekannt gemacht werden, siehe hierzu 9.1.1.3.

Nicht normativer Hinweis: Im Falle von „optimistischer" Locking-Strategie muss der Client darauf vorbereitet sein, bei Scheitern des Schreibens, den gesamten Zyklus des Holens der Daten und der Fortführung zu wiederholen[5].

9.1.7.1 wfs:LockFeature/@expiry

Das Attribut gibt an, nach wie vielen Minuten das Lock (vertreten durch das LockId) automatisch aufgegeben wird, falls es nicht vorher benutzt wurde. Der Defaultwert ist „5".

9.1.7.2 wfs:LockFeature/@lockAction

Der OKSTRA-WFS unterstützt die Werte ALL und SOME.

Im Falle ALL folgt eine Ausnahme, falls es nicht gelingt, alle selektierten Features mit einem Lock zu versehen. Bei SOME folgt eine Ausnahme nur, wenn kein Feature gelocked werden konnte.

Die Ausnahme bei SOME erfolgt nur, wenn das Nichtzustandekommen eines Locks auf vorliegende fremde Locks zurückgeht. Für den Fall, dass der Filter die leere Menge selektiert, soll ein *WFS_LockFeatureResponse* erzeugt werden, der weder *FeaturesLocked* noch *FeaturesNotLocked* enthält.

[5] Das muss er im Grunde auch bei „pessimistischen" Locks. Diese können z.B. durch „time-out" infolge des Überschreitens des expiry-Zeitraums verloren gehen.

Im Falle einer „optimistischen" Lockingstrategie ist es immer möglich, Locks für alle selektierten Features anzufordern. Die *lockAction* "SOME" / "ALL" hat in diesem Zusammenhang keine Bedeutung, da es niemals zu einer Ausnahme auf Grund von vorherigen Sperren kommen kann.

9.1.7.3 wfs:LockFeature/Query

Jedes Lock-Element führt entsprechend seiner Parametrisierung zu einer Suchoperation auf dem Datenbestand, die durch das Query-Element beschrieben wird.

Die gefundenen Features werden je nach lockAction (ALL|SOME) und Locking-Strategie („pessimistisch", „optimistische") mit Locks versehen. Werden infolge sich überschneidender Filter Feature-Exemplare mehrfach gelocked, so wird dies behandelt, als sei das Lock nur einmal vergeben.

9.1.7.4 wfs:LockFeature/wfs:Lock/fes:Filter

Das Filterelement, soll entsprechend den Angaben in 8.1.10 interpretiert werden.

9.1.7.5 wfs:WFS_LockFeatureResponse

Im Response-Dokument wird im Einzelnen (durch fes:ResourceId-Elemente) nachgewiesen, welche Features ein Lock erhalten haben und welche nicht.

Die Liste wfs:FeaturesNotLocked ist bei lockAction="ALL" naturgemäß immer leer und wird daher weggelassen.

Wird infolge des Filters kein Feature mit einem Lock versehen, so wird sowohl wfs:FeaturesLocked als auch wfs:FeaturesNotLocked weggelassen.

9.1.8 Operation: Transaction

Die Operation Transaction führt schreibende Operationen auf den FeatureTypes aus, für die dies gestattet ist. Welche dies im Einzelnen sind, ist im Capabilities-Response-Dokument beschrieben, siehe 9.1.1.4.

Ein OKSTRA-WFS gestattet es, vor verändernden Operationen Locks auf den betroffenen Features zu nehmen. Neben der im OGC-WFS vorgesehenen „pessimistischen" Locking-Strategie ist auch eine „optimistische" Strategie vorgesehen. Siehe hierzu die Ausführungen in 9.1.7.

Grundsätzlich werden die Einzeloperationen einer Transaction-Operation (also Insert, Update, Replace und Delete) in der angegebenen Reihenfolge ausgeführt. Die gesamte Operation erfolgt (unabhängig von der Existenz möglicher Locks) in jeder Sicht von außerhalb atomar und isoliert, d.h.:

- Parallele lesende und schreibende Operationen sehen, solange eine Transaction durchgeführt wird, grundsätzlich den Zustand vor der Transaction oder den Zustand nach der Transaction (keine „dirty reads", keine „lost updates"),
- Die Einzeloperationen gelingen entweder ganz oder gar nicht. D.h. wenn die Transaction fehlschlägt, ist der Zustand vor der Transaction-Operation wieder hergestellt.

Wenn ein Lock zur Verfügung steht und angegeben wird, so wird für die Features, für die das Lock gilt, sichergestellt, dass sie nicht durch andere Transactions verändert wurden. Bei „pessimistischer" Locking-Strategie wird dies erzwungen, indem das Feature für andere Transactions gesperrt wird, bei „optimistischer" Strategie, wird die eigene Transaction zurückgewiesen, falls eine Veränderung an den betroffenen Features festgestellt wird.

Veränderte Feature-Instanzen werden grundsätzlich von ihrem Lock befreit. Für die restlichen Features, die durch dasselbe Lock gesperrt sind, gibt es eine Option, das Lock zu behalten oder aufzulösen.

9.1.8.1 wfs:Transaction/@releaseAction

Ein OKSTRA-WFS unterstützt beide Werte, ALL und SOME, in der vom OGC-WFS-Standard beschriebenen Art und Weise.

9.1.8.2 wfs:Transaction/wfs:LockId

Falls in einer Transaction LockId nicht angegeben ist, so können mit dieser Transaction alle Features verändert werden, die kein fremdes Lock tragen. Der Versuch einer Veränderung an einem solchen Feature führt zu Abbruch und Rollback der Transaction. Im Falle einer „optimistischen" Lock-Strategie können Features mit Locks verändert werden – der Abbruch erfolgt dann später.

Wenn ein Lock zur Verfügung steht und angegeben wird, so wird für die Features, für die das Lock gilt, sichergestellt, dass sie nicht durch andere Transactions verändert wurden. Bei „pessimistischer" Locking-Strategie wird dies erzwungen, indem das Feature für andere Transactions gesperrt wird, bei „optimistischer" Strategie, wird die eigene Transaction zurückgewiesen, falls eine Veränderung an den betroffenen Features festgestellt wird.

Auch wenn ein Lock angegeben ist, darf der OSKTRA-WFS Features, welche kein fremdes Lock tragen, verändern.

Für die „optimistische" Strategie ist die LockId immer der Zeitstempel, der bei Aufruf der LockFeature- bzw. GetFeatureWithLock-Operation ermittelt wird. Wenn Features verändert werden (Delete oder Update) wird überprüft, ob die betroffenen Features zu einem Zeitpunkt nach diesem übergebenen Zeitstempel verändert wurden. Falls dies der Fall ist, wird die Transaktion abgebrochen. Wenn nicht, hat kein anderer Prozess diese Daten verändert und die Transaktion kann ausgeführt werden.

9.1.8.3 wfs:Transaction/wfs:Insert

Insert erzeugt neue Feature-Instanzen, aus

- einzelnen Features, gegeben in OKSTRA®-XML, oder
- aus einer gml:FeatureCollection, solche Features enthaltend.

Um relationale Beziehungen zwischen Features herstellen zu können, die in einer Transaction zusammen erzeugt werden, muss der OKSTRA-WFS lokale Verweise verarbeiten können. Dabei wird das gml:id-Attribut eines einzufügenden Features mit einem lokal eindeutigen Identifier versehen, das dann aus später auftretenden xlink:href-Properties heraus referiert werden kann. Der WFS setzt dann automatisch dafür die im Datenbestand erzeugten Identifikatoren ein. Ein OKSTRA-WFS muss derartige rückwärtsgerichtete lokale Verweise innerhalb eines Transaction-Dokuments verarbeiten können. Es wird empfohlen, auch Verweise in Vorwärtsrichtung zu unterstützen.

Beispiel:

```
<wfs:Transaction>
  <wfs:Insert>
    <okstra:Netzknoten gml:id="marke1">
      ...
    </okstra:Netzknoten>
    <okstra:Nullpunkt>
      ...
      <okstra:in_Netznoten xlink:href="#marke1"/>
    </okstra:Nullpunkt>
  </wfs:Insert>
</wfs:Transaction>
```

9.1.8.4 wfs:Transaction/wfs:Insert/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuordnung von Exceptions zu den einzelnen Bestandteilen der Transaction. Die Implementierung ist verpflichtend.

9.1.8.5 wfs:Transaction/wfs:Insert/@inputFormat

Das Akzeptieren des Formats „application/xml+gml;version=3.2.1" ist verpflichtend.

9.1.8.6 wfs:Transaction/wfs:Insert/@srsName

Die Implementierung dieses Attributs ist verpflichtend. Es definiert das Default-Koordinatenreferenzsystem für die im Insert-Inhalt angegebenen Geometrien. Dieses kann durch explizite Angaben bei den Geometrien der Features oder FeatureCollections überschrieben werden.

9.1.8.7 wfs:Transaction/wfs:Replace

Die Replace-Elemente ersetzen selektierte Features durch neue.

Die Selektion erfolgt über einen Filterausdruck. Dieser darf einen Datenhaltungszustand voraussetzen, wie er durch die Interpretation der Transaction bis vor das Replace-Element erreicht wurde.

9.1.8.8 wfs:Transaction/wfs:Replace/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuordnung von Exceptions zu den einzelnen Bestandteilen der Transaction. Die Implementierung ist verpflichtend.

9.1.8.9 wfs:Transaction/wfs:Replace/@inputFormat

Das Akzeptieren des Formats „application/xml+gml;version=3.2.1" ist verpflichtend.

9.1.8.10 wfs:Transaction/wfs:Replace/@srsName

Die Implementierung dieses Attributs ist verpflichtend. Es definiert das Default-Koordinatenreferenzsystem für die im Insert-Inhalt angegebenen Geometrien. Dieses kann durch explizite Angaben bei den Geometrien der Features oder FeatureCollections überschrieben werden.

9.1.8.11 wfs:Transaction/wfs:Replace/fes:Filter

Dieses verpflichtende Element beschreibt, welche Features durch die in der Operation mitgelieferten zu ersetzen sind.

9.1.8.12 wfs:Transaction/wfs:Update

Die Update-Elemente verändern die Properties von selektierten Features.

Die Selektion erfolgt über einen Filterausdruck. Dieser darf einen Datenhaltungszustand voraussetzen, wie er durch die Interpretation der Transaction bis vor das Update-Element erreicht wurde.

9.1.8.13 wfs:Transaction/wfs:Update/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuordnung von Exceptions zu den einzelnen Bestandteilen der Transaction. Die Implementierung ist verpflichtend.

9.1.8.14 wfs:Transaction/wfs:Update/@typeName

Das Attribut gibt den FeatureType an, dessen Feature-Instanzen verändert werden sollen, z.B. okstra:Abschnitt.

9.1.8.15 wfs:Transaction/wfs:Update/@inputFormat

Das Akzeptieren des Formats „application/xml+gml;version=3.2.1" ist verpflichtend.

Ein anderes Format ist im Fall Update nicht sinnvoll und soll deshalb nicht angeboten werden.

9.1.8.16 wfs:Transaction/wfs:Update/@srsName

Die Implementierung dieses Attributs ist nicht verpflichtend.

Das Koordinatensystem geht bereits aus den Geometrien in Property/Value hervor.

9.1.8.17 wfs:Transaction/wfs:Update/wfs:Property

In den Property-Angaben wird jeweils der Name eines Property seinem neuen Wert gegenübergestellt, und zusätzlich wird mit updateAction angegeben, wie die Änderung durchzuführen ist.

updateAction kann sein:

- insertBefore fügt vor das durch die ValueReference angegebene Element ein neues ein.
- insertAfter fügt nach dem durch die ValueReference angegebenem Element ein neues ein.
- replace ersetzt das durch die ValueReference angegebene Element.
- delete ersetzt das durch die ValueReference angegebene Element.

```
<wfs:Property>
    <wfs:Name>propertyName</wfs:Name>
    <wfs:Value>neuer Wert</wfs:Value>
</wfs:Property>
```

Das Setzen relationaler Bezüge erfolgt durch

<wfs:Value xlink:href="*id*"/>

wobei *id* eine URL wie in 8.1.9 beschrieben ist.

Der Wert von xlink:href darf auf den lokalen Bezeichner im gml:id eines vorangehenden, durch Insert erzeugten Features verweisen. Damit können relationale Bezüge zu Features hergestellt werden, welche in derselben Transaction erzeugt wurden.

Es wird empfohlen, auch Vorwärtsverweise auf von Insert erzeugte Features zuzulassen, die erst später im Transaction-Dokument aufgeführt sind.

9.1.8.18 wfs:Transaction/wfs:Update/fes:Filter

Das Filterelement soll entsprechend den Angaben in 8.1.10 interpretiert werden.

Fehlt es, so werden alle Instanzen des FeatureTypes verändert.

9.1.8.19 wfs:Transaction/wfs:Delete

Die Delete-Elemente löschen Features. Ihre relationalen Bezüge werden automatisch gepflegt. Etwaige resultierende Verstöße gegen Kardinalitätsbedingungen werden erst nach Abarbeiten der gesamten Transaction geprüft und führen im Fehlerfall zum Scheitern der Transaktion.

Die Selektion erfolgt über einen Filterausdruck. Dieser darf einen Datenhaltungszustand voraussetzen, wie er durch die Interpretation der Transaction bis vor das Delete-Element erreicht wurde.

9.1.8.20 wfs:Transaction/wfs:Delete/@handle

Die Angabe des handle-Attributs dient zur verbesserten Zuordnung von Exceptions zu den einzelnen Bestandteilen der Transaction. Die Implementierung ist verpflichtend.

9.1.8.21 wfs:Transaction/wfs:Delete/@typeName

Das Attribut gibt den FeatureType an, dessen Feature-Instanzen gelöscht werden sollen, z.B. okstra:Abschnitt.

9.1.8.22 wfs:Transaction/wfs:Delete/fes:Filter

Das Filterelement, soll entsprechend den Angaben in 8.1.10 interpretiert werden.

Fehlt es, so werden alle Instanzen des FeatureTypes gelöscht.

9.1.8.23 wfs:Transaction/wfs:Native

Native muss erkannt werden, muss aber nicht beachtet werden. Die gesamte Funktionalität des OKSTRA-WFS muss ohne Verwendung des Native-Elements erreichbar sein.

9.1.8.24 wfs:TransactionResponse

Das TransactionResponse-Dokument soll wie im OGC-WFS-Standard gefordert implementiert werden.

Die Angabe des TransactionResult-Elements ist nicht erforderlich, da ein OKSTRA-WFS immer atomare Transactions unterstützt.

9.1.9 Ausnahmebehandlung

Es gelten für den OKSTRA-WFS transaktional die Festlegungen aus 8.1.10. Zusätzlich werden die folgenden exceptionCodes festgelegt:

exceptionCode="..."	**Bedeutung**
InvalidLockId	Das in einer Transaction verwendete LockId ist nicht bekannt oder „expired".
FeatureCannotBeLocked	Auf wenigstens einem der verlangten Features konnte kein Lock genommen werden, weil es bereits anderweitig „gelockt" ist.
LockedFeatureCannotBeAltered	Die Transaction mit dem Versuch, ein Feature zu ändern oder zu löschen, welches durch eine fremdes Lock gesperrt ist, wird abgewiesen.
FeatureAlteredAfterLocking	Nur bei „optimistischem Locking": Die Transaction mit dem Versuch, ein Feature zu ändern oder zu löschen, welches nach der Locknahme verändert wurde, wird abgewiesen.
DataConsistencyFault	Der Versuch eines Verstoßes gegen das Datenschema durch eine Transaction wird abgewiesen.

9.2 Profil des Filter Encoding

Identisch zu 8.2.

9.3 Einschränkungen im OKSTRA®-Schema

Identisch zu 8.3.

9.4 Allgemeine zusätzliche Forderungen

9.4.1 Performanz

Es wird empfohlen, dass ein OKSTRA-WFS wenigstens 50 Features pro Sekunde einfügen/ersetzen kann. Ein ähnlicher Durchsatz ist für das Update und Delete einzeln adressierter Features zu erreichen.

9.4.2 Größenbeschränkungen

Keine zusätzlichen Anforderungen zu 8.4.2.

10 Authentifizierung und Autorisierung

10.1 Authentifizierung

Die folgenden Abschnitte beschreiben, welche Authentifizierungsverfahren für OKSTRA-WFS einsetzbar sind. Ein normativer Teil legt fest, welche Bedingungen hierfür erfüllt werden müssen.

Zu klären ist die Frage, wie die Außenwelt mitbekommt, welcher Authentifizierungsmechanismus von einem OKSTRA-WFS unterstützt wird.

Für Dienste, die nur innerhalb einer geschlossenen Organisation erreichbar sind, kann das organisatorisch festgelegt werden.

Für extern sichtbare Dienste könnte man die erlaubten Authentifizierungsarten z.B. im Capabilities-Dokument publizieren, dies müsste dann aber ohne Identitätsprüfung erreichbar sein.

Die hier dargelegten Ausführungen beziehen sich nur auf OKSTRA-WFS, bei denen die Authentifizierung Bestandteil der Implementation ist. Architekturen, die mit vorgeschalteten Proxies arbeiten, bleiben unberührt. In diesen Fällen sind die Mechanismen relevant, die die Proxy-Software anbietet.

10.1.1 http-basierte Authentifizierung

10.1.1.1 Basic Authentication

Die Basic Authentication nach RFC 2617 ist die häufigste Art der HTTP-Authentifizierung. Der Webserver fordert mit

```
WWW-Authenticate: Basic realm="RealmName"
```

eine Authentifizierung an, wobei RealmName eine Beschreibung des geschützten Bereiches darstellt. Der Browser sucht daraufhin nach Benutzername/Passwort für diese URL und fragt gegebenenfalls den Benutzer. Anschließend sendet er die Authentifizierung mit dem Authorization-Header in der Form Benutzername:Passwort Base64-codiert an den Server. Beispiel:

```
Authorization: Basic d2lraTpwZWRpYQ==
```

d2lraTpwZWRpYQ== ist die Base64-Codierung von wiki:pedia und steht damit für Benutzername wiki, Passwort pedia. Ein Nachteil dieses Verfahrens ist, dass Benutzername und Passwort nur aus technischen Gründen codiert, jedoch nicht verschlüsselt werden. Aus sicherheitstechnischer Sicht ist dieses Verfahren daher genauso unsicher als würde das Passwort im Klartext übertragen werden. Bei einer Verschlüsselung mit SSL/TLS bei HTTPS wird bereits vor der Übermittlung des Passwortes eine verschlüsselte Verbindung aufgebaut, so dass auch bei Basic Authentication das Passwort nicht abhörbar ist. (Quelle: Wikipedia, https://de.wikipedia.org/wiki/HTTP-Authentifizierung)

Für nicht-interaktive Clients und SSO-Umgebungen muss der Authorization-Header vom Client transparent ohne menschliche Interaktion erzeugt werden.

Damit das Passwort nicht gestohlen werden kann, muss die Verbindung mit https verschlüsselt aufgebaut werden.

10.1.1.2 Digest Access Authentication

Bei der Digest Access Authentication (ebenfalls RFC 2617) sendet der Server zusammen mit dem WWW-Authenticate-Header eine eigens erzeugte zufällige Zeichenfolge (Nonce). Der Browser berechnet den Hashcode (in der Regel MD5) einer Kombination aus Benutzername, Passwort, erhaltener Zeichenfolge, HTTP-Methode und angeforderter URI. Diese sendet er im Authorization-

Header zusammen mit dem Benutzernamen und der zufälligen Zeichenfolge zurück an den Server. Dieser berechnet seinerseits die Prüfsumme und vergleicht. Das Verfahren ist damit dem des Message Authentication Code ähnlich.

Vorausgesetzt die benutzte Hashfunktion ist kryptographisch sicher, nützt ein Abhören der Kommunikation einem Angreifer nichts, da sich durch die Nutzung einer Hashfunktion die Zugangsdaten nicht rekonstruieren lassen und durch die Nutzung der Nonce für jede Anforderung anders lauten. (Speziell wird die weit verbreitete Hashfunktion MD5 nicht mehr als sicher erachtet.) Die restliche Datenübertragung ist jedoch nicht geschützt. Um dies zu erreichen, kann etwa HTTPS verwendet werden. (Quelle: Wikipedia, https://de.wikipedia.org/wiki/HTTP-Authentifizierung).

Unterstützt ein OKSTRA-WFS diese Form der Authentifizierung, muss ein sicherer Hashcode verwendet werden, mindestens SHA-1.

10.1.2 Authentifizierung mit SOAP

Die Authentifizierung über SOAP erfordert die Implementierung von Teilen von WS-Security. Diese werden hier zunächst kurz beschrieben.

WS-Security ist ein Standard aus dem Kontext der WS-*-Spezifikationen. Im Wesentlichen beschreibt er ein Kommunikationsprotokoll, das es ermöglicht, Sicherheitsaspekte bei Webservices zu berücksichtigen. Am 19. April 2004 wurde der OASIS-Standard in seiner Version 1.0 veröffentlicht und am 1. Februar 2006 auf die Version 1.1 aktualisiert (Referenz 4.4).

Der Standard beinhaltet Spezifikationen, die vorschreiben, wie Nachrichtenintegrität und -verschlüsselung im Rahmen von Webservices sichergestellt werden können. Dabei schreibt WS-Security jedoch nicht alle Details vor, sondern setzt vielmehr auf bereits bestehende „Verfahren" auf (XML-Signature und XML-Encryption).

WS-Security beinhaltet drei Hauptmechanismen:

- die Möglichkeit, *Securitytokens* als Teil der SOAP-Nachricht zu übertragen,
- Signierung von Nachrichten und
- Verschlüsselung von Nachrichten.

Dabei wird genau vorgeschrieben, wo und wie Signaturen, Verschlüsselungsinformationen sowie besagte Securitytokens in der SOAP-Nachricht eingefügt werden müssen. Für die Security Token erlaubt der Standard folgende Formate:

- User Name Token. Das Token enthält wenigstens einen Benutzernamen.
- Binary Security Token. Das Token enthält binär kodierte Daten, die die Authentifizierungsinformation enthalten.
- XML Security Token. Das Token enthält ein XML-Fragment. Der Inhalt ist nicht a priori festgelegt, sondern muss durch ein WS-Security-Profil näher beschrieben werden (z.B. SAML, siehe unten).
- Encrypted Data Token. Dies ist kein eigenes Tokenformat, sondern wird mit einem der drei vorgenannten verwendet, die hierbei verschlüsselt übertragen werden.

Zusätzlich zum WS-Security-Basisstandard werden folgende WS-Security-Profile zur Erstellung des SecurityTokens für häufig anzutreffende Authentifizierungsmechanismen unterschieden:

- Benutzername (Username/Password)
- X.509-Zertifikat

- Kerberos-Ticket
- SAML-Assertion
- REL-Dokument

(Quelle: Wikipedia https://de.wikipedia.org/wiki/WS-Security)

Das X.509-Zertifikat-Profil verwendet verschlüsselte digitale Ausweise, die von einer vertrauenswürdigen Ausgabestelle ausgestellt werden. Es sollte in OKSTRA-WFS-Diensten eingesetzt werden, die intern in oder zwischen Organisationen arbeiten, die bereits eine X.509-basierende Authentifizierungsarchitektur haben. Achtung: Die Clients müssen das Profil auch beherrschen.

Das Kerberos-Ticket-Protokoll verwendet sogenannte Tickets. Im Gegensatz zu X.509 sind dies digitale Ausweise, die nur für die Dauer einer Session gültig sind. Es kann in OKSTRA-WFS-Diensten eingesetzt werden, die in Organisationen arbeiten, deren Authentifizierungssystem auf Kerberos basiert und die Single-Sign-On anbieten wollen, z.B. Active Directory in Windows-Umgebungen.

Beim SAML-Protokoll wird eine Assertion (Zusicherung) im Security-Header übertragen, die in der Security Assertion Markup Language (SAML) abgefasst ist. Eine Assertion enthält kurz gesagt Information WER WANN VON WEM WOMIT authentifiziert wurde. Zusätzlich können noch Bedingungen angegeben werden, z.B. Gültigkeitszeitraum.

Neben den soeben genannten Authentifizierungsverfahren gibt es weitere, die in z.B. offenen Single-Sign-On-Lösungen wie Jasig-CAS Verwendung finden. Sollen solche auch für OKSTRA-WFS einsetzbar sein, ohne dass eine Umsetzung auf die genannten Standard-Profile erfolgt, müssen entsprechende WS-Security-Profile speziell für den OKSTRA® neu geschaffen werden. Im Falle von CAS könnte z.B. ein CAS-Service-Ticket in einem Binary Security Token übergeben werden. Zur Umsetzung dieser Strategie müsste jedoch zuerst eine Liste der von den Bundesländern gewünschten Verfahren erstellt werden.

10.1.2.1 User-Passwort-Authentifizierung

Das Profil ist beschrieben in https://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf.

Bei dieser Form der Authentifizierung wird im SOAP-Security-Header ein UsernameToken übertragen. Dieses enthält:

- den Username
- das Passwort (optional), entweder als Text oder als sogenannter Digest
- das Nonce (optional), eine zufällig generierte Zeichenkette, falls das Passwort als Digest vorliegt
- die Zeitmarke der Entstehung des Tokens (optional)

10.1.2.2 X.509-Authentifizierung

Das Profil ist beschrieben in http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf.

Hierbei wird ein Zertifikat nach dem internationalen Standard X.509 im SOAP-Security-Header als BinarySecurityToken übertragen. Für einen OKSTRA-WFS eignet sich die Übertragung eines Einzelzertifikates am besten (ValueType #X509v3). Das Zertifikat übermittelt die Identifikation der Stelle, die den OKSTRA-WFS nutzen möchte. Zur Authentifizierung muss dieser die digitale Signatur des Zertifikates prüfen. Der Signatur-Algorithmus ist im Zertifikat angegeben, es muss also lediglich eine Softwareprozedur aktiviert werden, der man das Zertifikat zur Prüfung übergibt (z.B. java.security.cert oder Windows Certificate API).

10.1.2.3 Kerberos-Authentifizierung

Das Profil ist beschrieben in http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-KerberosTokenProfile.pdf.

Diese Form der Authentifizierung eignet sich z.B. gut für Active Directory Domains, um Single Sign On zu realisieren. Eine Einführung gibt z.B. http://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/single-signon.html. Dort ist beschrieben, wie man ein Ticket behandeln muss, um die Identität des Users der Session herauszubekommen.

10.1.2.4 SAML-Authentifizierung

Das Profil ist beschrieben in https://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf.

Die Authentifizierung per SAML geschieht über sog. Assertions, das sind in XML kodierte „Ausweise", die Attribute des Nutzers der aktuellen Session nachweisen, sowie die Information enthalten, wer die Identität des aktuellen Users wie überprüft hat, wie lange der Ausweis gültig ist usw.

SAML ist eine für GDI-DE vorgesehene Technologie, deswegen sollten OKSTRA-WFS dieses Profil wahlweise unterstützen können.

10.1.2.5 REL-Dokument

Rights Enforcement Language (REL) ist keine offene Technologie, deswegen sollen OKSTRA-WFS dieses Profil nicht unterstützen.

10.1.3 Anforderungen (normativ)

Authentifizierende OKSTRA-WFS müssen eine oder mehrere der Methoden 10.1.1.1 - 10.1.1.2, 10.1.2.1 - 10.1.2.4 unterstützen.

Falls sowohl http- als auch SOAP-basierte Authentifizierung zusammen implementiert werden, gibt es Seiteneffekte, da für SOAP-Nachrichten über http dessen Authentifizierung auch greift (bei der Verhandlung über die Übermittlung der Credentials weiß der Service nicht, dass ein SOAP-Paket ankommen wird.) Authentifizierende OKSTRA-WFS dürfen deshalb entweder nur über die http/https-WFS-Schnittstelle oder nur über die SOAP-WFS-Schnittstelle verfügen. Falls beide Schnittstellen erforderlich sind, müssen zwei WFS eingerichtet werden, die auf gemeinsame Geschäftslogik zugreifen.

10.1.3.1 http Basic Authentication

Als Protokoll muss https verwendet werden.

10.1.3.2 http Digest Access Authentication

Es muss ein sicherer Hashcode verwendet werden, mindestens SHA-1.

10.1.3.3 SOAP User-Password-Authentifizierung

Drei Fälle sind zu unterscheiden:

- Nur Username: Dieses Verfahren soll nur verwendet werden, wenn die Authentifizierung durch ein dem OKSTRA-WFS vorgeschalteten Authentifizierungs-Proxy erfolgt.

- Username plus Text-Passwort: In diesem Fall muss die Übertragung verschlüsselt erfolgen. Dies kann entweder durch Übertragung des SOAP-Requests per https gewährleistet werden oder durch Verschlüsselung des Tokens im SOAP-Header (EncryptedDataToken). Die zweite Möglichkeit erfordert den Einsatz einer zwischen den Clients und dem OKSTRA-WFS abgestimmten Verschlüsselungstechnik.

- Username plus Digest: Hierzu wird zuerst ein Nonce (zufällige Zeichenkette) erzeugt. Das Passwort, die Zeitmarke und das Nonce werden konkateniert, mit SHA-1 (oder besser) gehasht und dann als Passwort-Digest im UsernameToken versendet. Der OKSTRA-WFS bezieht aus seiner Benutzerdatenbank das Passwort, kombiniert es mit den im UsernameToken übermittelten Nonce und Zeitstempel und vergleicht das Resultat mit dem Digest aus dem UsernameToken.

10.1.3.4 SOAP X509-Authentifizierung

Es sind nur Einzelzertifikate (Value Type X509v3) zulässig.

10.1.3.5 SOAP Kerberos-Authentifizierung

Keine zusätzlichen Anforderungen.

10.1.3.6 SOAP SAML-Authentifizierung

Es sind Assertions gemäß SAML 1.1 oder SAML 2.0 zulässig. Die übermittelte Assertion muss ein SAML 1.1 AuthenticationStatement oder SAML 2.0 AuthnStatement enthalten.

10.2 Autorisierung

10.2.1 Begriffsbestimmungen

Autorisierung bedeutet die Überprüfung der Berechtigung zum Zugriff durch eine (Zugriffs-)Aktion eines Benutzers auf eine Ressource, die von einem Server bereitgestellt werden kann, kurz gefasst; WER DARF WAS WOMIT. Gesteuert wird dies durch *Berechtigungsregeln*.

10.2.1.1 Benutzer

Ein Benutzer kann eine Person sein, die mit dem Server über eine User Interface interagiert (z.B. Browser) oder aber eine als Service arbeitende Softwarekomponente.

Der Benutzer wird hierbei zunächst über einen Namen, den Benutzernamen, identifiziert. Die Verfahren zur Identitätsprüfung von Benutzern sind im Abschnitt 10.1 über Authentifizierung dargelegt. Ein Autorisierungssystem geht davon aus, dass diese Prüfung mit positivem Ergebnis erfolgt ist.

10.2.1.2 Gruppen

Falls ein Informationssystem vielen Benutzern zugänglich ist, wird eine Autorisierung über Benutzernamen schnell unpraktisch, weil dann für jeden Benutzer einzeln sein Berechtigungsprofil im System hinterlegt werden muss. Der übliche Ausweg ist die Bildung von Gruppen von Benutzern, deren Berechtigungsprofile identisch sind. Eine ausgezeichnete Gruppe ist die Gruppe ALLE, die alle Benutzer umfasst, die sich dem System gegenüber identifizieren können.

10.2.1.3 Rechteinhaber

Rechteinhaber sind die Einheiten, denen die Berechtigungen zugeteilt werden. Ein Rechteinhaber kann entweder eine Benutzer oder eine Gruppe sein.

10.2.1.4 Klassen

Der Begriff Klassen wird hier synonym zu FeatureType verwendet.

10.2.1.5 Aktionen

10.2.1.5.1 Klassenaktionen

Als Aktionen bieten sich die Aufrufe der in der Tabelle des Kapitels 6 genannten Operationen zum Lesen und Schreiben von Fachdaten an. Dabei ist es zusätzlich sinnvoll, die Unteroperationen von Transaction getrennt als Aktionen im Sinne der Autorisierung aufzufassen.

Die Unterdrückung von GetCapabilities und DescribeFeatureType unterbindet de facto die Nutzbarkeit eines WFS, auf diese sind daher Berechtigungsregeln nicht anwendbar, jedoch kann die Response dieser Operationen von den Berechtigungsregeln abhängig sein. Da auch ListStoredQueries und DescribeStoredQuery Metadaten nachweisen, verfährt man für diese zweckmäßig analog.

Jeder Aufruf einer Stored Query gilt ebenfalls als Aktion im Sinne der Autorisierung.

Somit gibt es für die Autorisierung folgende Aktionen:

Autorisierung	
Aktion	Beschreibung
GetPropertyValue	Diese Operation erfragt den Wert einer einzelnen Eigenschaft eines Features mit Hilfe eines Filter-Ausdrucks.
GetFeature	Die Operation erfragt die Inhalte eines oder mehrerer Features. Die Auswahl der Features erfolgt anhand des FeatureTypes und (optional) eines Filter-Ausdrucks, der es gestattet, Features nach inhaltlichen Kriterien zu selektieren. Die Features werden ausgedrückt in GML (Geography Markup Language) zurückgebracht.
LockFeature	Durch diese Operation werden „Locks" auf durch Filter selektierte Features genommen. Features, auf denen ein Lock besteht, können nicht durch andere Clients verändert werden. Es können auch keine weiteren Locks für diese genommen werden.
GetFeatureWithLock	Wie GetFeature. Die Operation nimmt aber zusätzlich ein „Lock" auf den ausgewählten Features, das sicherstellt, dass diese bis zu einer nachfolgenden Transaction-Operation, die sich auf das Lock bezieht, nicht durch andere Clients verändert werden können.
Transaction	In einer Transaction können beliebig viele Features erzeugt, verändert oder gelöscht werden. Eine Transaction wirkt als Einheit, d.h. ganz oder gar nicht. Sollen Features mit „Locks" an einer Transaction teilnehmen, so muss die zugehörige „Lock-Id" angegeben werden.
Transaction.Insert	Einfügen von Features
Transaction.Update	Modifizieren von Features
Transaction.Delete	Löschen von Features
Transaction.Replace	Ersetzen von Features
StoredQuery *name*	Standardabfragen

10.2.1.5.2 Eigenschaftsoperationen

Zusätzlich zu den Aktionen, die auf ganze Features wirken, müssen auch solche betrachtet werden, die auf Eigenschaften (GML-Properties) wirken:

- ReadValue: Wert auslesen,
- ReplaceValue: Überschreiben eines Wertes,
- AddValue: Hinzufügen von Werten (Kardinalität 0..n),
- DeleteValue: Entfernen von Werten.

10.2.2 Regeltypen für Berechtigungen

Es werden in diesem Abschnitt Typen für atomare Berechtigungsregeln angeführt. Die Berechtigung zum Zugriff ergibt sich durch Verrechnung der atomaren Berechtigungsregeln für denselben Rechteinhaber, die dieselbe Klasse, Eigenschaft und Aktion betreffen. Das Autorisierungssystem muss Mechanismen bereitstellen, die die Reihenfolge bei der Verrechnung eindeutig regelt, z.B. Prioritäten oder Operatoren analog zum Java &&-Operator.

Regeltyp 1.1: Von einem Rechteinhaber R darf die Aktion A angewendet werden.

Eine derartige Regel steuert den Zugriff global für alle Klassen, die der OKSTRA-WFS bereitstellen kann. R ist ein Benutzer oder eine Benutzergruppe. Ist R nicht angegeben, gilt die Regel für ALLE. A ist eine der Aktionen der Tabelle in 10.2.1.5.1.

Regeltyp 1.2: Auf Features der Klasse K darf von einem Rechteinhaber R die Aktion A angewendet werden.

Eine derartige Regel steuert den Zugriff für die Klassen K, die der OKSTRA-WFS bereitstellen kann. K ist eine OKSTRA®-Objektart. Falls K nicht instanzierbar ist und daher Unterklassen hat, gilt die Regel auch für die Unterklassen. Für instanzierbare Klassen gilt die Regel nur für die angegebene Klasse.

Regeltyp 1.3: Auf Features der Klasse K darf unter der Bedingung B von einem Rechteinhaber R die Aktion A angewendet werden.

Wie Regeltyp 1.2, aber mit Instanzenselektion („row-level-security"). Die Bedingung B ist ein Filterausdruck gemäß Filter Encoding 2.0.

Regeltyp 2.1: Auf die Eigenschaft E der Klasse K darf von einem Rechteinhaber R die Aktion A angewendet werden.

Eine derartige Regel steuert den Zugriff auf Eigenschaften von Klassen K, die der OKSTR-WFS bereitstellen kann. K ist eine OKSTRA®-Objektart. A ist eine der Aktionen aus 10.2.1.5.2.

Regeltyp 2.2: Auf die Eigenschaft E der Klasse K darf unter der Bedingung B von einem Rechteinhaber R die Aktion A angewendet werden.

Wie Regeltyp 2.1, aber mit Instanzenselektion („row-level-security"). Die Bedingung B ist ein Filterausdruck gemäß Filter Encoding 2.0.

10.2.3 Anforderungen (normativ)

Anforderung	Begründung
Die Implementierung eines Autorisierungssystems muss „wasserdicht" sein, d.h. die Durchsetzung der Berechtigungsregeln muss für alle möglichen Zugriffspfade gewährleistet sein.	Von einem WFS können nicht nur die explizit in einem Query benannten Features nachgewiesen werden, sondern auch solche, die implizit über Pfadausdrücke oder die Auflösungstiefe (@resolveDepth) erreicht werden.
Falls die Nutzung einer Klasse durch die Berechtigungsregeln vollständig ausgeschlossen ist, sollen die Operationen GetCapabilities und DescribeFeatureType diesen FeatureType nicht nachweisen.	Dem Benutzer soll nur das durch ihn nutzbare Schema nachgewiesen werden.
Falls die Nutzung einer Eigenschaft durch die Berechtigungsregeln vollständig ausgeschlossen ist, soll die Operation DescribeFeatureType nicht nachweisen.	Dem Benutzer soll nur das durch ihn nutzbare Schema nachgewiesen werden.
Falls die Nutzung einer Operation durch die Be-	Dem Benutzer sollen nur die durch ihn nutzba-

rechtigungsregeln vollständig ausgeschlossen ist, soll diese Operation nicht in den Operation-Metadata von GetCapabilities nachgewiesen werden.	ren Funktionen nachgewiesen werden.
Falls die Nutzung eines Stored Query durch die Berechtigungsregeln vollständig ausgeschlossen ist, sollen die Operationen ListStoredQueries und DescribeStoredQuery diese Stored Query nicht nachweisen.	Dem Benutzer sollen nur die durch ihn nutzbaren Funktionen nachgewiesen werden.
Das Autorisierungssystem muss Mechanismen bereitstellen, die die Reihenfolge bei der Verrechnung der Berechtigungsregeln eindeutig regeln.	Abhängig von der Reihenfolge der Auswertung könnten sich bei der Verrechnung der Berechtigungen unterschiedliche Resultate ergeben.
Das Autorisierungssystem soll row-level-security unterstützen.	Es können sonst keine geometrischen Einschränkungen durchgesetzt werden.

10.2.4 Architektur

Das Autorisierungssystem ist nicht Bestandteil der Schnittstelle eines OKSTRA-WFS, daher kann dieses Standardisierungsdokument keine Vorgaben für die Implementierung machen, sondern nur Anforderungen aussprechen und Alternativen untersuchen. Insbesondere kann das Autorisierungssystem vollständig innerhalb des OKSTRA-WFS und seiner Datenhaltung implementiert werden oder teilweise über evtl. vorhandene Externsysteme.

Zweckmäßig für die folgenden Ausführungen sind die Begriffe (aus dem XACML-Standard übernommen):

- Policy Decision Point (PDP), das ist der Teil eines Autorisierungssystems, der die Berechtigungsentscheidung anhand der Berechtigungsregeln ausrechnet,
- Policy Enforcement Point (PEP), das ist der Teil, der die Berechtigungsentscheidung umsetzt, also z.B. die über die WFS-Schnittstelle übermittelten Querys mit einem Autorisierungsquery kombiniert,
- Policy Administration Point (PAP), das ist der Teil, mit dem die Berechtigungsregeln formuliert und gespeichert werden.

Für eine wasserdichte und performante Autorisierungslösung muss der PEP möglichst in bzw. nahe an die Datenhaltung gelegt werden, um unerwünschte Features und Eigenschaften erst gar nicht weiter verarbeiten zu müssen.

Falls der PDP in eine Komponente außerhalb des OKSTRA-WFS verlagert wird, müssen die Parameter E, K, B, A, R der Berechtigungsregeln vom WFS an den PDP über eine Schnittstelle übertragen werden. Dies setzt voraus, dass der WFS-Query soweit vorverarbeitet ist, dass alle betroffenen E und K bekannt sind (A und R sind unmittelbar im WFS-Request abzulesen). Der PDP rechnet dann für alle (E,K,A,R)-Tupel eine Ja-Nein-Entscheidung für jedes E und K unter Verwendung der in der Berechtigungsregel enthaltenen Bedingung B aus, und überträgt diese zurück. In dem Fall eines externen PDP ist auch der PAP sinnvollerweise extern.

Falls nur der PAP in eine Komponente außerhalb des OKSTRA-WFS verlagert wird, müssen die Berechtigungsregeln selbst über eine Schnittstelle in diesen übertragen werden. Dazu ist eine Anfrage an den PAP zu richten, die mindestens R beinhaltet.

Die Standardisierung von Schnittstellen zwischen einem OKSTRA-WFS und externen Autorisierungskomponenten sind nicht Bestandteil dieser Spezifikation. Sie sind abhängig von den nutzbaren Externprodukten, sollten jedoch organisationsweit einheitlich sein.

11 Metadaten

Mittlerweile haben alle Bundesländer für ihre Geodateninfrastrukturen ein Metadatenprofil erstellt. Die Empfehlung dieser Spezifikation ist daher, für die OKSTRA-WFS-Dienste die vom Metadatenprofil des jeweiligen Bundeslandes geforderten Pflichtmetadaten für Daten und Dienste zu erfassen.